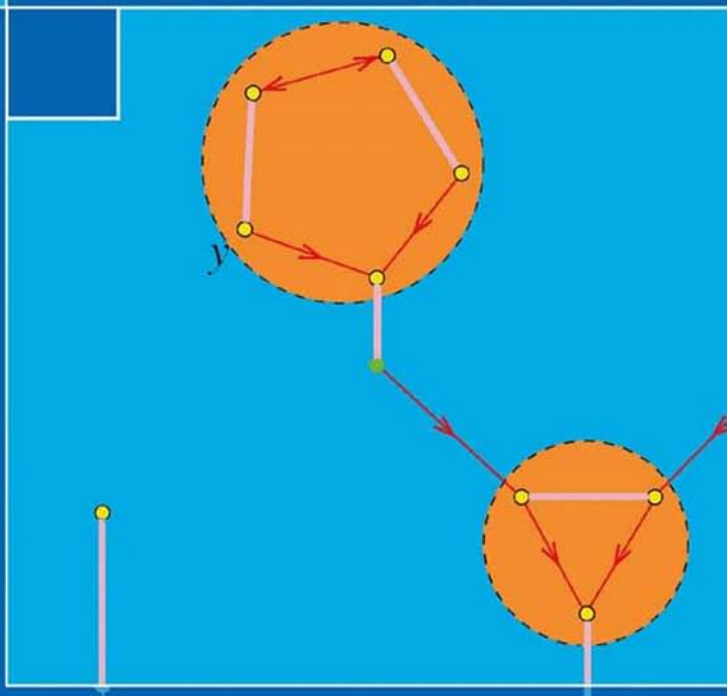
Bernhard Korte
Jens Vygen

Collection IRIS
dirigée par Nicolas Puech



Optimisation combinatoire

Théorie et algorithmes

Traduit de l'anglais
par Jean Fonlupt et Alexandre Skoda

Springer

Optimisation combinatoire

Springer
Paris
Berlin
Heidelberg
New York
Hong Kong
Londres
Milan
Tokyo

Bernhard Korte
Jens Vygen

Optimisation combinatoire

Théorie et algorithmes

Traduit de l'anglais par Jean Fonlupt et Alexandre Skoda

Springer

Bernhard Korte
Research Institute
for Discrete Mathematics
University of Bonn
Lennéstraße 2
53113 Bonn
Germany
dm@or.uni-bonn.de

Jens Vygen
Research Institute
for Discrete Mathematics
University of Bonn
Lennéstraße 2
53113 Bonn
Germany
vygen@or.uni-bonn.de

Traducteurs

Jean Fonlupt
Professeur émérite
Université Paris-VI
Faculté de mathématiques
175, rue du Chevaleret
75013 Paris
Jean.Fonlupt@math.jussieu.fr

Alexandre Skoda
Université Paris-I
Panthéon-Sorbonne
Centre d'économie de la Sorbonne
106-112, boulevard de l'Hôpital
75013 Paris
alexandre.skoda@univ-paris1.fr

ISBN : 978-2-287-99036-6 Springer Paris Berlin Heidelberg New York

Maquette de couverture : Jean-François MONTMARCHÉ
Illustration de couverture : Ina PRINZ

Collection *IRIS*
Dirigée par Nicolas Puech

Ouvrages parus :

– *Méthodes numériques pour le calcul scientifique*. *Programmes en Matlab*
A. Quarteroni, R. Sacco, F. Saleri, Springer-Verlag France, 2000

– *Calcul formel avec MuPAD*
F. Maltey, Springer-Verlag France, 2002

– *Architecture et micro-architecture des processeurs*
B. Goossens, Springer-Verlag France, 2002

– *Introduction aux mathématiques discrètes*
J. Matousek, J. Nesetril, Springer-Verlag France, 2004

– *Les virus informatiques : théorie, pratique et applications*
É. Filiol, Springer-Verlag France, 2004

– *Introduction pratique aux bases de données relationnelles*. *Deuxième édition*
A. Meier, Springer-Verlag France, 2006

– *Bio-informatique moléculaire. Une approche algorithmique*
P.A. Pevzner, Springer-Verlag France, 2006

– *Algorithmes d'approximation*
V. Vazirani, Springer-Verlag France, 2006

– *Techniques virales avancées*
É. Filiol, Springer-Verlag France, 2007

– *Codes et turbocodes*
C. Berrou, Springer-Verlag France, 2007

– *Introduction à Scilab. Deuxième édition*
J.P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikouhah, S. Steer,
Springer-Verlag France, 2007

– *Maple : règles et fonctions essentielles*
N. Puech, Springer-Verlag France, 2009

– *Les virus informatiques : théorie, pratique et applications. Deuxième édition*
É. Filiol, Springer-Verlag France, 2009

À paraître :

– *Concepts et méthodes en phylogénie moléculaire*
G. Perrière, Springer-Verlag France, 2010

Préface

Ce livre est la traduction française de la quatrième édition du livre *Combinatorial Optimization : Theory and Algorithms* écrit par deux éminents spécialistes, Bernhard Korte et Jens Vygen, professeurs à l'université de Bonn. Considéré comme un ouvrage de référence, il s'adresse à des chercheurs confirmés qui travaillent dans le champ de la recherche fondamentale ou de ses applications (R&D). Il donne une vision complète de l'optimisation combinatoire et peut donc aussi intéresser de nombreux scientifiques non spécialistes ayant une bonne culture en mathématiques et des connaissances de base en informatique.

L'optimisation combinatoire est un domaine assez récent des mathématiques appliquées, qui plonge ses racines dans la combinatoire (principalement la théorie des graphes), la recherche opérationnelle et l'informatique théorique. Une des raisons de son développement est liée au nombre considérable de problèmes concrêts qu'elle permet de formuler. Il s'agit en grande partie de problèmes pour lesquels on connaît de «bons» algorithmes de résolution ; ceux-ci sont étudiés dans la première partie de ce livre. Une des originalités de cet ouvrage, par rapport à d'autres traités, est de présenter les algorithmes de résolution ayant la meilleure borne de complexité connue à ce jour.

La seconde partie traite des problèmes difficiles à résoudre sur le plan algorithmique et connus sous le nom de problèmes *NP-difficiles*. Le plus célèbre d'entre eux, celui du voyageur de commerce, fait l'objet, au chapitre 21, d'une étude particulièrement approfondie. D'autres tout aussi importants, comme les problèmes de conception de réseaux, de multi-flots, de localisation de services, etc., bénéficient également d'une présentation détaillée, ce qui est peu fréquent dans la littérature et mérite d'être signalé.

Dans la traduction que nous proposons, nous avons cherché à traduire en français toutes les expressions et tous les termes anglo-saxons même quand aucune traduction n'existait ; il y a cependant quelques exceptions pour des termes très techniques qui ne sont universellement connus que sous leur dénomination anglaise. Nous avons en outre inclus quelques améliorations et corrections écrites par les auteurs après la parution de l'édition originale actuelle ; celles-ci seront intégrées dans la cinquième édition anglaise, actuellement en préparation.

Paris, juillet 2009 *Jean Fonlupt et Alexandre Skoda*

Avant-propos à la quatrième édition originale

Avec quatre éditions anglaises et quatre traductions en cours, nous sommes très heureux de l'évolution de notre livre ; celui-ci a été révisé, actualisé et amélioré de manière significative pour cette quatrième édition. Nous y avons inclus des matières classiques, parfois manquantes dans les éditions précédentes, notamment sur la programmation linéaire, la méthode network simplex et le problème de la coupe maximum. Nous avons également ajouté de nouveaux exercices et mis à jour les références.

Nous sommes reconnaissants à l'Union des académies allemandes des sciences et des lettres et à l'Académie des sciences du Land Rhénanie-du-Nord-Westphalie pour leur soutien permanent par l'intermédiaire du projet «Mathématiques discrètes et applications». Nous remercions également pour leurs commentaires précieux tous ceux qui nous ont contacté après la troisième édition, en particulier Takao Asano, Christoph Bartoschek, Bert Besser, Ulrich Brenner, Jean Fonlupt, Satoru Fujishige, Marek Karpinski, Jens Maßberg, Denis Naddef, Sven Peyer, Klaus Radke, Rabe von Randow, Dieter Rautenbach, Martin Skutella, Markus Struzyna, Jürgen Werber, Minyi Yue, et Guochuan Zhang. Nous continuerons à fournir des informations actualisées sur cet ouvrage à l'adresse :
`http ://www.or.uni-bonn.de/~vygen/co.html`

Bonn, août 2007 *Bernhard Korte et Jens Vygen*

Sommaire

Chapitre 1

Introduction

Commençons cet ouvrage par deux exemples.

Une machine est utilisée pour percer des trous dans des plaques de circuits imprimés. Comme de nombreux circuits sont produits, il est souhaitable que chaque circuit soit fabriqué aussi rapidement que possible. Nous ne pouvons agir sur le temps de perçage de chaque trou qui est fixé, mais nous pouvons chercher à minimiser le temps total de déplacement de la perceuse. Habituellement, les perceuses effectuent des déplacements dans deux directions : la table se déplace horizontalement tandis que le bras de la machine se déplace verticalement. Comme ces deux mouvements peuvent se faire simultanément, le temps nécessaire pour ajuster la machine entre deux positions est proportionnel au maximum des distances horizontales et verticales parcourues. Cette quantité est souvent appelée distance de la norme infini. (Les vieilles machines ne peuvent se déplacer que dans une direction à la fois ; le temps d'ajustement est alors proportionnel à la 1-distance, somme des distances horizontale et verticale.)

Un parcours optimal pour le perçage est donné par un ordre des positions des trous $p_1, \ldots, p_n$ qui rend minimum la quantité $\sum_{i=1}^{n-1} d(p_i, p_{i+1})$, d étant la distance de la norme infini : si $p = (x, y)$ et $p' = (x', y')$ sont deux points du plan, alors $d(p, p') := \max\{|x - x'|, |y - y'|\}$. Un ordre des trous peut être représenté par une permutation, c.-à-d. une bijection $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$.

La meilleure permutation dépend bien entendu de la position des trous ; pour chaque ensemble de positions, nous aurons une instance spécifique (suivant l'usage, nous utiliserons le terme «instance» de préférence à «exemple»). Nous dirons qu'une instance est une liste de points du plan, c.-à-d. une liste des coordonnées des trous à percer. Le problème peut alors se formuler de la manière suivante :

PROBLÈME DE PERÇAGE	
Instance	Un ensemble de points $p_1, \ldots, p_n \in \mathbb{R}^2$.
Tâche	Trouver une permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ telle que $\sum_{i=1}^{n-1} d(p_{\pi(i)}, p_{\pi(i+1)})$ soit minimum.

Décrivons maintenant notre deuxième exemple. Nous devons effectuer un ensemble de tâches dont nous connaissons les temps d'exécution. Chaque tâche peut être confiée à une partie des employés. Plusieurs employés peuvent être affectés à une même tâche et chaque employé peut travailler sur plusieurs tâches mais pas simultanément. Notre objectif est d'exécuter l'ensemble des tâches aussi rapidement que possible.

Dans ce modèle, il suffira de déterminer le temps d'affectation de chaque employé aux différentes tâches. Le temps d'exécution de l'ensemble des tâches est alors égal au temps de travail de l'employé le plus occupé. Nous devons donc résoudre le problème suivant :

PROBLÈME D'AFFECTATION DES TÂCHES

Instance Un ensemble de nombres $t_1, \ldots, t_n \in \mathbb{R}_+$ (les temps d'exécution des n tâches), un nombre $m \in \mathbb{N}$ d'employés, et un sous-ensemble non vide $S_i \subseteq \{1, \ldots, m\}$ d'employés pour chaque tâche $i \in \{1, \ldots, n\}$.

Tâche Trouver des nombres $x_{ij} \in \mathbb{R}_+$ pour tout $i = 1, \ldots, n$ et $j \in S_i$ tels que $\sum_{j \in S_i} x_{ij} = t_i$ pour $i = 1, \ldots, n$ et tel que $\max_{j \in \{1,\ldots,m\}} \sum_{i : j \in S_i} x_{ij}$ soit minimum.

Voilà deux exemples typiques de problèmes d'optimisation combinatoire. La manière de modéliser un problème pratique en un problème abstrait d'optimisation combinatoire n'est pas l'objet de ce livre ; il n'y a d'ailleurs aucune recette pour réussir dans cette démarche. Outre la précision des données et des résultats attendus, il est souvent important pour un modèle d'ignorer certains paramètres non significatifs (par exemple, le temps de perçage qui ne peut être optimisé ou l'ordre suivant lequel les employés exécutent les tâches).

Notons enfin qu'il ne s'agit pas de résoudre un cas particulier d'un problème, comme celui du perçage ou celui d'affectation des tâches, mais de résoudre tous les cas possibles de ces problèmes. Étudions d'abord le PROBLÈME DE PERÇAGE.

1.1 Énumération

Quelle est l'allure d'une solution du PROBLÈME DE PERÇAGE ? Ce problème a un nombre infini d'instances possibles (tout ensemble fini de points du plan) et nous ne pouvons donc faire la liste des permutations optimales associées à toutes les instances. Ce que nous recherchons, c'est un algorithme qui associe, à chaque instance, une solution optimale. Un tel algorithme existe : étant donné un ensemble de n points, calculer la longueur du chemin associé à chacune des $n!$ permutations.

Il y a de nombreuses manières de formuler un algorithme, la différence se faisant principalement par le niveau de détails ou par le langage formel utilisé. Nous n'accepterons pas la proposition suivante comme définissant un algorithme : étant donné un ensemble de n points, trouver un chemin optimal qui sera l'output, c.-à-d. le résultat, car rien n'est dit sur la manière de trouver la solution optimale. La sugges-

tion précédente, d'énumérer l'ensemble des $n!$ permutations, est plus utile à condition de préciser la manière d'énumérer ces permutations. Voici une méthode : énumérons par comptage tous les n-uplets des nombres $1, \dots, n$, c.-à-d. les n^n vecteurs de $\{1, \dots, n\}^n$: partons de $(1, \dots, 1, 1)$, $(1, \dots, 1, 2)$ jusqu'à $(1, \dots, 1, n)$, passons à $(1, \dots, 1, 2, 1)$, et ainsi de suite : à chaque étape, nous ajoutons 1 à la dernière composante sauf si celle-ci vaut n, auquel cas nous revenons à la dernière composante plus petite que n, lui ajoutons 1 et réinitialisons à 1 toutes les composantes suivantes. Cette technique est parfois appelée «*backtracking*» (en français, retour arrière). L'ordre selon lequel les vecteurs de $\{1, \dots, n\}^n$ sont énumérés est appelé ordre lexicographique.

Définition 1.1. *Soient $x, y \in \mathbb{R}^n$ deux vecteurs. Nous dirons qu'un vecteur x est* **lexicographiquement plus petit** *que y s'il existe un indice $j \in \{1, \dots, n\}$ tel que $x_i = y_i$ pour $i = 1, \dots, j-1$ et $x_j < y_j$.*

Il nous suffit maintenant de vérifier si, au cours de l'énumération, chaque vecteur de $\{1, \dots, n\}^n$ a des composantes différentes et voir dans ce cas si le chemin représenté par cette permutation est plus court que le meilleur chemin trouvé précédemment.

Comme cet algorithme énumère n^n vecteurs, il nécessitera au moins n^n étapes. Cela n'est pas très efficace puisque le nombre de permutations de $\{1, \dots, n\}$ est $n!$ qui est bien plus petit que n^n. (Par la formule de Stirling $n! \approx \sqrt{2\pi n}\frac{n^n}{e^n}$ (Stirling [1730]) ; voir exercice 1.) Montrons comment énumérer tous les chemins en approximativement $n^2 \cdot n!$ étapes grâce à l'algorithme suivant qui énumère toutes les permutations suivant un ordre lexicographique :

ALGORITHME D'ÉNUMÉRATION DES CHEMINS

Input	Un nombre naturel $n \geq 3$. Un ensemble $\{p_1, \dots, p_n\}$ de points dans le plan.
Output	Une permutation $\pi^* : \{1, \dots, n\} \to \{1, \dots, n\}$ telle que $\mathit{coût}(\pi^*) := \sum_{i=1}^{n-1} d(p_{\pi^*(i)}, p_{\pi^*(i+1)})$ soit minimum.

① $\pi(i) := i$ et $\pi^*(i) := i$ pour $i = 1, \dots, n$. Posons $i := n - 1$.

② Soit $k := \min(\{\pi(i) + 1, \dots, n + 1\} \setminus \{\pi(1), \dots, \pi(i-1)\})$.

③ **If** $k \leq n$ **then** :
  $\pi(i) := k$.
  If $i = n$ et $\mathit{coût}(\pi) < \mathit{coût}(\pi^*)$ **then** $\pi^* := \pi$.
  If $i < n$ **then** $\pi(i+1) := 0$ et $i := i + 1$.
If $k = n + 1$ **then** $i := i - 1$.
If $i \geq 1$ **then go to** ②.

Partant de $(\pi(i))_{i=1,\dots,n} = (1, 2, 3, \dots, n-1, n)$ et $i = n - 1$, l'algorithme trouve à chaque étape la meilleure valeur possible suivante de $\pi(i)$ (sans utiliser $\pi(1), \dots, \pi(i-1)$). S'il n'existe plus aucune possibilité pour $\pi(i)$ (c.-à-d. $k =$

$n+1$), alors l'algorithme décrémente i (*backtracking*). Sinon il affecte à $\pi(i)$ la nouvelle valeur k. Si $i = n$, la nouvelle permutation est évaluée, sinon l'algorithme évaluera toutes les valeurs possibles pour $\pi(i+1), \ldots, \pi(n)$, en affectant à $\pi(i+1)$ la valeur 0 et en incrémentant i.

Ainsi tous les vecteurs de permutation $(\pi(1), \ldots, \pi(n))$ sont générés suivant un ordre lexicographique. Par exemple, les premières itérations dans le cas $n = 6$ sont décrites comme suit :

$$\begin{array}{llll}
 & \pi := (1,2,3,4,5,6), & i := 5 & \\
k := 6, & \pi := (1,2,3,4,6,0), & i := 6 & \\
k := 5, & \pi := (1,2,3,4,6,5), & & \textit{coût}(\pi) < \textit{coût}(\pi^*)\,? \\
k := 7, & & i := 5 & \\
k := 7, & & i := 4 & \\
k := 5, & \pi := (1,2,3,5,0,5), & i := 5 & \\
k := 4, & \pi := (1,2,3,5,4,0), & i := 6 & \\
k := 6, & \pi := (1,2,3,5,4,6), & & \textit{coût}(\pi) < \textit{coût}(\pi^*)\,?
\end{array}$$

Puisque l'algorithme compare le coût de la solution courante à π^*, le meilleur chemin actuel, il fournit bien le chemin optimal. Mais quel est le nombre d'étapes ? La réponse dépendra de ce que nous appelons un «pas» de l'algorithme. Comme le nombre de pas ne doit pas dépendre de l'implémentation, nous devons ignorer les facteurs constants. Ainsi, ① nécessitera au moins $2n+1$ étapes et au plus cn étapes, c étant une constante. La notation suivante sera utile pour ignorer les constantes :

Définition 1.2. *Soient $f, g : D \to \mathbb{R}_+$ deux fonctions. Nous dirons que f est $O(g)$ (et nous écrirons parfois $f = O(g)$) s'il existe des constantes $\alpha, \beta > 0$ telles que $f(x) \leq \alpha g(x) + \beta$ pour tout $x \in D$. Si $f = O(g)$ et $g = O(f)$ nous dirons alors que $f = \Theta(g)$ (et bien entendu $g = \Theta(f)$). Dans ce cas, f et g auront le même* **taux de croissance**.

Remarquons que la relation $f = O(g)$ n'implique aucune symétrie entre f et g. Pour illustrer cette définition, prenons $D = \mathbb{N}$ et soit $f(n)$ le nombre de pas ou d'étapes élémentaires de ①. En posant $g(n) = n$ ($n \in \mathbb{N}$), il est évident que $f = O(g)$ (et que, également, $f = \Theta(g)$) ; nous dirons que ① s'exécute en un temps $O(n)$ ou en temps linéaire. L'exécution de ③ se fait en un nombre constant de pas (nous dirons aussi en temps $O(1)$ ou en temps constant) sauf dans le cas où les coûts de deux chemins doivent être comparés, ce qui nécessitera un temps $O(n)$.

Que peut-on dire de ② ? Vérifier si $j = \pi(h)$ pour tout $j \in \{\pi(i)+1, \ldots, n\}$ et tout $h \in \{1, \ldots, i-1\}$ se fait en $O((n - \pi(i))i)$ étapes, c.-à-d. en un temps $\Theta(n^2)$. On peut améliorer ce temps en utilisant un tableau auxiliaire indexé par $1, \ldots, n$:

② **For** $j := 1$ **to** n **do** $aux(j) := 0$.
For $j := 1$ **to** $i-1$ **do** $aux(\pi(j)) := 1$.
$k := \pi(i) + 1$.
While $k \leq n$ et $aux(k) = 1$ **do** $k := k + 1$.

De cette manière, ② s'exécute en un temps $O(n)$. Nous n'étudierons pas dans ce livre ce genre d'améliorations algorithmiques, laissant au lecteur le choix des bonnes mises en œuvre.

Examinons maintenant le temps total d'exécution de l'algorithme. Puisque le nombre de permutations est $n!$, il nous faut trouver le temps de calcul entre deux permutations. Le compteur i peut décroître de la valeur n à un indice i', une nouvelle valeur de $\pi(i') \leq n$ étant trouvée. Puis le compteur est réincrémenté jusqu'à la valeur $i = n$. Tant que le compteur i est constant, chacune des étapes ② et ③ est exécutée une seule fois, sauf dans le cas $k \leq n$ et $i = n$; dans ce cas ② et ③ sont exécutées deux fois. Ainsi le nombre de pas entre deux permutations est au plus $4n$ fois ② et ③, c.-à-d. $O(n^2)$. Le temps total d'exécution de l'ALGORITHME D'ÉNUMÉRATION DES CHEMINS est $O(n^2 n!)$.

On peut faire encore mieux ; une analyse plus fine montre que le temps de calcul est seulement $O(n \cdot n!)$ (exercice 4).

Cependant, le temps de calcul de l'algorithme est trop important quand n devient grand, car le nombre de chemins croît d'une manière exponentielle avec le nombre de points ; déjà pour 20 points, on a $20! = 2\ 432\ 902\ 008\ 176\ 640\ 000 \approx 2,4 \cdot 10^{18}$ chemins différents et même les ordinateurs les plus puissants auraient besoin de plusieurs années pour tous les examiner. Ainsi une énumération complète est impossible à envisager même pour des instances de taille modeste.

L'objet de l'optimisation combinatoire est de trouver de meilleurs algorithmes pour ce type de problèmes. Nous devrons souvent trouver le meilleur élément d'un ensemble fini de solutions réalisables (dans nos exemples : chemins de perçage ou permutations). Cet ensemble n'est pas défini explicitement, mais dépend implicitement de la structure du problème. Un algorithme doit pouvoir exploiter cette structure.

Dans le PROBLÈME DE PERÇAGE une instance avec n points sera décrite par $2n$ coordonnées. Alors que l'algorithme précédent énumère les $n!$ chemins, on peut imaginer qu'il existe un algorithme trouvant le chemin optimal plus rapidement, disons en n^2 étapes de calcul. On ne sait pas si un tel algorithme existe (on verra cependant au chapitre 15 que cela est improbable). Il existe cependant des algorithmes bien meilleurs que ceux fondés sur la méthode d'énumération.

1.2 Temps d'exécution des algorithmes

On peut donner une définition formelle d'un algorithme, et c'est ce que nous ferons au chapitre 15.1. Cependant, de tels modèles conduisent à des descriptions longues et fastidieuses. Il en est de même pour les preuves mathématiques : bien que le concept de preuve puisse être formalisé, personne n'utilise un tel formalisme pour décrire des preuves, car elles deviendraient trop longues et presque illisibles.

Ainsi les algorithmes présentés dans ce livre seront-ils écrits dans un langage informel. Cependant, ils seront suffisamment détaillés pour qu'un lecteur ayant un peu d'expérience puisse les programmer sur un ordinateur sans trop d'effort.

Puisque nous ne prenons pas en compte les facteurs constants quand nous mesurons le temps de calcul, nous n'avons pas à spécifier un modèle concret d'ordinateur. Nous comptons les pas élémentaires sans nous soucier du temps d'exécution de ces pas. Comme exemples de pas élémentaires, citons les affectations de variables,

l'accès aléatoire à une variable dont l'adresse est stockée dans un autre registre, les sauts conditionnels (if – then – go to), ainsi que les opérations arithmétiques élémentaires telles que l'addition, la soustraction, la multiplication, la division, la comparaison de nombres.

Un algorithme consiste en un ensemble d'inputs valides et une suite d'instructions composées d'opérations élémentaires, de telle sorte que pour chaque input valide, le déroulement de l'algorithme soit une suite bien définie d'opérations élémentaires fournissant un output. La question essentielle sera alors d'obtenir une borne satisfaisante du nombre d'opérations, en fonction de la taille de l'input.

L'input est en général une liste de nombres. Si tous ces nombres sont des entiers, nous pouvons les coder dans une représentation binaire en réservant un emplacement de $O(\log(|a|+2))$ bits pour stocker un entier a. Les nombres rationnels peuvent être stockés en codant séparément leur numérateur et leur dénominateur. La **taille de l'input** notée $\text{taille}(x)$ d'une instance x avec des données rationnelles est le nombre total de bits utilisés dans la représentation binaire.

Définition 1.3. *Soit A un algorithme qui accepte des inputs d'un ensemble X, et soit $f : \mathbb{N} \to \mathbb{R}_+$. S'il existe une constante $\alpha > 0$ telle que A se termine après au plus $\alpha f(\text{taille}(x))$ pas élémentaires (en incluant les opérations arithmétiques) pour chaque input $x \in X$, nous dirons alors que A* **s'exécute en un temps** *$O(f)$. Nous dirons également que $O(f)$ est le* **temps de calcul** *ou la* **complexité** *de A.*

Définition 1.4. *Un algorithme acceptant des inputs rationnels est dit* **polynomial** *s'il s'exécute en un temps $O(n^k)$ quand la taille de l'input est n, k étant fixé, et si la taille de tous les nombres intermédiaires calculés n'excède pas $O(n^k)$ bits.*

Un algorithme acceptant des inputs arbitraires est dit **fortement polynomial** *si son temps de calcul est $O(n^k)$ pour tout input de n nombres, k étant une constante fixée, et s'il se termine en temps polynomial dans le cas d'inputs rationnels. Si $k = 1$, nous dirons que l'algorithme est* **linéaire**.

Notons que le temps de calcul peut être différent pour des instances distinctes de même taille (ce n'était pas le cas pour l'ALGORITHME D'ÉNUMÉRATION DES CHEMINS). Nous considérerons le temps de calcul dans le pire des cas, c.-à-d. la fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le maximum du temps de calcul d'une instance de taille n. Pour certains algorithmes, nous ne connaissons pas le taux de croissance de f, mais nous avons seulement une borne supérieure.

Il se peut que le temps de calcul dans le pire des cas soit une mesure pessimiste si le pire des cas se produit rarement. Dans certaines situations, un temps de calcul moyen fondé sur des modèles probabilistes serait plus adéquat, mais nous n'aborderons pas cette question dans ce livre.

Si A est un algorithme qui, pour chaque input $x \in X$, calcule l'output $f(x) \in Y$, nous dirons que A **calcule** $f : X \to Y$. Une fonction calculée par un algorithme polynomial sera dite **calculable en temps polynomial**.

Les algorithmes polynomiaux sont quelquefois appelés «bons» ou «efficaces». Ce concept a été introduit par Cobham [1964] et Edmonds [1965]. La table 1.1 illustre cela en fournissant les temps de calcul pour divers temps de complexité.

Table 1.1.

n	$100n \log n$	$10n^2$	$n^{3.5}$	$n^{\log n}$	2^n	$n!$
10	3 μs	1 μs	3 μs	2 μs	1 μs	4 ms
20	9 μs	4 μs	36 μs	420 μs	1 ms	76 années
30	15 μs	9 μs	148 μs	20 ms	1 s	$8 \cdot 10^{15}$ a.
40	21 μs	16 μs	404 μs	340 ms	1100 s	
50	28 μs	25 μs	884 μs	4 s	13 jours	
60	35 μs	36 μs	2 ms	32 s	37 années	
80	50 μs	64 μs	5 ms	1075 s	$4 \cdot 10^7$ a.	
100	66 μs	100 μs	10 ms	5 heures	$4 \cdot 10^{13}$ a.	
200	153 μs	400 μs	113 ms	12 années		
500	448 μs	2.5 ms	3 s	$5 \cdot 10^5$ a.		
1000	1 ms	10 ms	32 s	$3 \cdot 10^{13}$ a.		
10^4	13 ms	1 s	28 heures			
10^5	166 ms	100 s	10 années			
10^6	2 s	3 heures	3169 a.			
10^7	23 s	12 jours	10^7 a.			
10^8	266 s	3 années	$3 \cdot 10^{10}$ a.			
10^{10}	9 heures	$3 \cdot 10^4$ a.				
10^{12}	46 jours	$3 \cdot 10^8$ a.				

Pour différentes tailles d'inputs n, nous indiquons les temps de calcul de six algorithmes qui nécessitent $100n \log n$, $10n^2$, $n^{3.5}$, $n^{\log n}$, 2^n, et $n!$ opérations élémentaires ; nous supposons qu'une opération élémentaire s'effectue en une nanoseconde. Comme partout dans ce livre, «log» est le logarithme en base 2.

Ainsi que la table 1.1 le montre, les algorithmes polynomiaux sont plus rapides pour les instances de taille suffisamment importante. Cette table indique également que les facteurs constants de taille modérée ne sont pas très importants si on considère la croissance asymptotique du temps de calcul.

La table 1.2 indique la taille maximum d'inputs résolubles en une heure pour les six algorithmes précédents. Pour (a) nous supposons qu'une opération élémentaire s'effectue en une nanoseconde ; (b) donne les résultats pour une machine dix fois plus rapide. Les algorithmes polynomiaux peuvent traiter de grandes instances en des temps raisonnables. Cependant, même en multipliant par 10 la rapidité de calcul des ordinateurs, on n'augmente pas de manière significative la taille des instances que l'on peut résoudre pour des algorithmes exponentiels, ce qui n'est pas le cas pour les algorithmes polynomiaux.

Les algorithmes (fortement) polynomiaux et si possible linéaires sont ceux qui nous intéressent. Il existe des problèmes pour lesquels il n'existe aucun algorithme polynomial et d'autres pour lesquels il n'existe aucun algorithme. (Par exemple, un problème qui peut se résoudre en un temps fini mais pas en temps polynomial

Table 1.2.

	$100n \log n$	$10n^2$	$n^{3.5}$	$n^{\log n}$	2^n	$n!$
(a)	$1.19 \cdot 10^9$	60000	3868	87	41	15
(b)	$10.8 \cdot 10^9$	189737	7468	104	45	16

est celui de décider si une expression régulière définit l'ensemble vide ; voir Aho, Hopcroft et Ullman [1974]. Un problème pour lequel il n'existe aucun algorithme est le «HALTING PROBLEM», décrit dans l'exercice 1 du chapitre 15.)

Cependant, presque tous les problèmes étudiés dans ce livre appartiennent à une des deux classes suivantes : pour les problèmes de la première classe, il existe un algorithme polynomial ; pour les problèmes de la seconde, l'existence d'un algorithme polynomial est une question ouverte. Néanmoins, nous savons que si un de ces problèmes peut se résoudre en temps polynomial, alors tous les problèmes appartenant à cette seconde classe sont également résolubles en temps polynomial. Une formulation et une preuve de cette affirmation seront données au chapitre 15.

Le PROBLÈME DE L'AFFECTATION DES TÂCHES appartient à la première classe, le PROBLÈME DE PERÇAGE appartient à la seconde.

Ces deux classes divisent à peu près ce livre en deux parties. Nous étudierons d'abord les problèmes pour lesquels on connaît des algorithmes polynomiaux. Puis, à partir du chapitre 15, nous nous intéresserons aux problèmes difficiles. Bien qu'on ne connaisse aucun algorithme polynomial dans ce cas, il existe souvent de bien meilleures méthodes que l'énumération complète. De plus, pour de nombreux problèmes (incluant le PROBLÈME DE PERÇAGE), on peut trouver des solutions approchées à un certain pourcentage de l'optimum en temps polynomial.

1.3 Problèmes d'optimisation linéaire

Revenons sur notre deuxième exemple, le PROBLÈME D'AFFECTATION DES TÂCHES, pour illustrer brièvement un sujet central de ce livre.

Le PROBLÈME D'AFFECTATION DES TÂCHES est totalement différent du PROBLÈME DE PERÇAGE puisque chaque instance non triviale a un nombre infini de solutions. Nous pouvons reformuler ce problème en introduisant une variable T qui sera le temps nécessaire à l'achèvement de toutes les tâches :

$$\begin{array}{lll} \min & T & \\ \text{s.c.} & \sum_{j \in S_i} x_{ij} = t_i & (i \in \{1, \ldots, n\}) \\ & x_{ij} \geq 0 & (i \in \{1, \ldots, n\},\ j \in S_i) \\ & \sum_{i : j \in S_i} x_{ij} \leq T & (j \in \{1, \ldots, m\}) \end{array} \tag{1.1}$$

(s.c. est une abréviation pour «sous les contraintes»)

Les nombres t_i et les ensembles S_i ($i = 1, \ldots, n$) sont donnés, et nous cherchons à calculer les variables x_{ij} et T. Un problème d'optimisation de ce type, avec une fonction objectif linéaire et des contraintes linéaires, est appelé **programme linéaire**. L'ensemble des solutions réalisables de (1.1) est un **polyèdre** ; cet ensemble convexe a un nombre fini de points extrêmes qui inclut la **solution optimale** de ce programme linéaire. Un programme linéaire peut donc, en théorie, se résoudre par énumération complète, mais de bien meilleures méthodes existent comme nous le verrons ultérieurement.

Bien que de nombreux algorithmes existent pour résoudre des programmes linéaires, les techniques générales sont souvent moins performantes que les algorithmes spécifiques qui exploitent la structure du problème. Dans notre exemple, il est judicieux de modéliser les ensembles S_i, $i = 1, \ldots, n$, à l'aide d'un **graphe** : associons à chaque tâche i et à chaque employé j un point (appelé sommet) et relions par une arête un employé i et une tâche j si i peut être affecté à j (c.-à-d. si $j \in S_i$). Les graphes constituent une structure combinatoire fondamentale ; de nombreux problèmes d'optimisation combinatoire se décrivent de manière naturelle dans le contexte de la théorie des graphes.

Supposons que le temps d'exécution de chaque tâche soit de une heure et que nous voulions savoir si toutes les tâches seront terminées en une heure. Ce problème revient à trouver des nombres x_{ij} ($i \in \{1, \ldots, n\}, j \in S_i$) tels que $0 \leq x_{ij} \leq 1$ pour tout i et j, $\sum_{j \in S_i} x_{ij} = 1$ pour $i = 1, \ldots, n$, et $\sum_{i : j \in S_i} x_{ij} \leq 1$ pour $j = 1, \ldots, n$. On peut montrer que si ce problème a une solution, celle-ci peut être choisie entière, les quantités x_{ij} valant alors 0 ou 1. Cela revient à affecter chaque tâche à un seul employé qui effectuera au plus une seule tâche. Dans le langage de la théorie des graphes, nous cherchons un **couplage** couvrant toutes les tâches. Le problème de la recherche d'un couplage optimal est un des problèmes classiques de l'optimisation combinatoire.

L'étude et le rappel de notions de base en théorie des graphes et en programmation linéaire sera l'objet des chapitres 2 et 3. Au chapitre 4 nous montrerons comment résoudre les programmes linéaires en temps polynomial, et au chapitre 5 nous étudierons les polyèdres entiers. Les chapitres suivants seront consacrés à l'étude de problèmes classiques en optimisation combinatoire.

1.4 Tri

Concluons ce chapitre en nous intéressant à un cas particulier du PROBLÈME DE PERÇAGE ; plus précisément, nous supposerons que tous les trous doivent être percés sur une même ligne horizontale. Il suffit de connaître une seule coordonnée pour chaque point p_i, $i = 1, \ldots, n$. Une solution du problème de perçage est alors facile à trouver : il s'agit de faire le tri des points selon cette coordonnée, le bras de la machine se déplaçant alors de la gauche vers la droite. Nous n'aurons donc pas à examiner les $n!$ permutations, pour trouver le chemin optimal : il est en effet très facile de trier n nombres dans un ordre non décroissant en un temps $O(n^2)$.

Trier n nombres en un temps $O(n \log n)$ demande un peu plus de réflexion. Il y a de nombreux algorithmes ayant cette complexité ; nous présentons ici l'algorithme bien connu de TRI-FUSION (en anglais, *merge-sort*) : la liste initiale est d'abord divisée en deux sous-listes de même taille approximative. Puis chaque sous-liste est triée (récursivement, par le même algorithme). Enfin, les deux sous-listes triées sont fusionnées. Cette méthode, appelée «diviser pour régner» (*divide and conquer* en anglais) est souvent utilisée. Voir le paragraphe 17.1 pour une autre illustration.

Nous n'avons pas présenté ce qu'on appelle les algorithmes récursifs. Ce ne sera pas nécessaire ici ; il nous suffira de savoir que tout algorithme récursif peut être transformé en un algorithme séquentiel sans accroître le temps de calcul. Cependant, certains algorithmes sont plus faciles à formuler (et à implémenter) en utilisant la récursivité, et c'est ce que nous ferons quelquefois dans cet ouvrage.

ALGORITHME TRI-FUSION

Input Une liste $a_1, \ldots, a_n$ de nombres réels.

Output Une permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ telle que $a_{\pi(i)} \leq a_{\pi(i+1)}$ pour tout $i = 1, \ldots, n-1$.

① **If** $n = 1$ **then** $\pi(1) := 1$ et **stop** (**return** π).

② $m := \lfloor \frac{n}{2} \rfloor$.
Soit $\rho :=$TRI-FUSION$(a_1, \ldots, a_m)$.
Soit $\sigma :=$TRI-FUSION$(a_{m+1}, \ldots, a_n)$.

③ $k := 1, l := 1$.
While $k \leq m$ et $l \leq n - m$ **do** :
If $a_{\rho(k)} \leq a_{m+\sigma(l)}$ **then** $\pi(k + l - 1) := \rho(k)$ et $k := k + 1$
else $\pi(k + l - 1) := m + \sigma(l)$ et $l := l + 1$.
While $k \leq m$ **do** : $\pi(k + l - 1) := \rho(k)$ et $k := k + 1$.
While $l \leq n - m$ **do** : $\pi(k + l - 1) := m + \sigma(l)$ et $l := l + 1$.

Comme exemple, considérons la liste «69, 32, 56, 75, 43, 99, 28». L'algorithme divise d'abord cette liste en deux listes, «69, 32, 56» et «75, 43, 99, 28» puis trie récursivement chacune des deux sous-listes. Nous obtenons les deux permutations $\rho = (2, 3, 1)$ et $\sigma = (4, 2, 1, 3)$ correspondant aux listes triées «32, 56, 69» et «28, 43, 75, 99». Ces deux listes sont alors fusionnées de la manière suivante :

					$k := 1,$	$l := 1$
$\rho(1) = 2,$	$\sigma(1) = 4,$	$a_{\rho(1)} = 32,$	$a_{\sigma(1)} = 28,$	$\pi(1) := 7,$		$l := 2$
$\rho(1) = 2,$	$\sigma(2) = 2,$	$a_{\rho(1)} = 32,$	$a_{\sigma(2)} = 43,$	$\pi(2) := 2,$	$k := 2$	
$\rho(2) = 3,$	$\sigma(2) = 2,$	$a_{\rho(2)} = 56,$	$a_{\sigma(2)} = 43,$	$\pi(3) := 5,$		$l := 3$
$\rho(2) = 3,$	$\sigma(3) = 1,$	$a_{\rho(2)} = 56,$	$a_{\sigma(3)} = 75,$	$\pi(4) := 3,$	$k := 3$	
$\rho(3) = 1,$	$\sigma(3) = 1,$	$a_{\rho(3)} = 69,$	$a_{\sigma(3)} = 75,$	$\pi(5) := 1,$	$k := 4$	
	$\sigma(3) = 1,$		$a_{\sigma(3)} = 75,$	$\pi(6) := 4,$		$l := 4$
	$\sigma(4) = 3,$		$a_{\sigma(4)} = 99,$	$\pi(7) := 6,$		$l := 5$

Théorème 1.5. *L'*ALGORITHME TRI-FUSION *répond correctement et s'exécute en un temps* $O(n \log n)$.

Preuve. Il est évident que cet algorithme répond correctement. Si $T(n)$ est le temps de calcul (nombre de pas) sur des instances ayant n nombres, observons que $T(1) = 1$ et que $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 3n + 6$. (Les constantes dans l'expression $3n + 6$ dépendent de la manière dont est défini un pas de l'algorithme.)

Nous affirmons que cela implique que $T(n) \leq 12n \log n + 1$. Le cas $n = 1$ étant trivial, nous procéderons par induction. Pour $n \geq 2$, en supposant que l'inégalité est vraie pour $1, \ldots, n-1$, nous avons

$$\begin{aligned} T(n) &\leq 12 \left\lfloor \frac{n}{2} \right\rfloor \log \left(\frac{2}{3} n \right) + 1 + 12 \left\lceil \frac{n}{2} \right\rceil \log \left(\frac{2}{3} n \right) + 1 + 3n + 6 \\ &= 12n(\log n + 1 - \log 3) + 3n + 8 \\ &\leq 12n \log n - \frac{13}{2} n + 3n + 8 \;\leq\; 12n \log n + 1, \end{aligned}$$

parce que $\log 3 \geq \frac{37}{24}$. □

Cet algorithme s'applique aussi au tri d'éléments d'un ensemble totalement ordonné, pourvu que l'on puisse comparer deux éléments quelconques en temps constant. Peut-il exister un algorithme plus rapide, disons linéaire ? Si on ne peut trouver l'ordre qu'à la suite de comparaisons successives de deux éléments, il est possible de montrer que tout algorithme nécessite au moins $\Theta(n \log n)$ comparaisons dans le pire des cas. En effet, on peut représenter le résultat d'une comparaison par zéro ou un. Le résultat de toutes les comparaisons est donc une chaîne binaire (une suite de zéro et de un). Deux ordres différents pour l'input de l'algorithme produisent deux chaînes binaires différentes (sinon on ne pourrait distinguer ces deux ordres). Pour un input ayant n éléments, il y a donc $n!$ ordres possibles et $n!$ chaînes binaires susceptibles d'être produites. Comme le nombre de chaînes binaires de longueur plus petite que $\lfloor \frac{n}{2} \log \frac{n}{2} \rfloor$ est $2^{\lfloor \frac{n}{2} \log \frac{n}{2} \rfloor} - 1 < 2^{\frac{n}{2} \log \frac{n}{2}} = (\frac{n}{2})^{\frac{n}{2}} \leq n!$, le nombre nécessaire de comparaisons est au moins $\frac{n}{2} \log \frac{n}{2} = \Theta(n \log n)$.

On voit donc que le temps de calcul de l'ALGORITHME TRI-FUSION est optimal à un facteur constant près. On peut cependant trier des entiers ou des chaînes suivant un ordre lexicographique grâce à des algorithmes linéaires ; voir l'exercice 7. Han [2004] a proposé un algorithme pour trier n entiers en $O(n \log \log n)$.

Il y a très peu de problèmes pour lesquels des bornes inférieures non triviales de ce type existent. On aura souvent besoin d'un minorant de l'ensemble des opérations élémentaires pour obtenir une borne inférieure superlinéaire.

Exercices

1. Montrer que pour tout $n \in \mathbb{N}$:

$$e \left(\frac{n}{e} \right)^n \leq n! \leq en \left(\frac{n}{e} \right)^n .$$

Indication : utiliser la relation $1 + x \leq e^x$ pour tout $x \in \mathbb{R}$.

2. Montrer que $\log(n!) = \Theta(n \log n)$.
3. Montrer que $n \log n = O(n^{1+\epsilon})$ pour tout $\epsilon > 0$.
4. Montrer que le temps de calcul de l'ALGORITHME D'ÉNUMÉRATION DES CHEMINS est $O(n \cdot n!)$.
5. Soit un algorithme dont le temps de calcul est $\Theta(n(t + n^{1/t}))$, n étant la taille de l'input et t un paramètre positif arbitraire. Comment choisir t en fonction de n pour que le temps de calcul qui est une fonction de n ait un taux de croissance minimum ?
6. Soient s, t deux chaînes binaires de longueur m. Nous dirons que s est lexicographiquement plus petite que t s'il existe un indice $j \in \{1, \ldots, m\}$ tel que $s_i = t_i$ pour $i = 1, \ldots, j-1$ et $s_j < t_j$. Soient alors n chaînes de longueur m que nous souhaitons trier suivant un ordre lexicographique. Montrer qu'on peut résoudre ce problème en un temps $O(nm)$.
   Indication : regrouper les chaînes selon leur premier bit et trier chaque groupe.
7. Proposer un algorithme qui trie une liste de nombres naturels $a_1, \ldots, a_n$, c.-à-d. qui trouve une permutation π telle que $a_{\pi(i)} \leq a_{\pi(i+1)}$ $(i = 1, \ldots, n-1)$ en un temps $O(\log(a_1 + 1) + \cdots + \log(a_n + 1))$.
   Indication : trier d'abord les chaînes codant les entiers suivant leur longueur. Appliquer ensuite l'algorithme de l'exercice 6.
   Note : l'algorithme étudié ici et dans l'exercice précédent est quelquefois appelé le tri radix.

Références

Littérature générale :

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. [2001] : Introduction to Algorithms. Second Edition. MIT Press, Cambridge 2001

Knuth, D.E. [1968] : The Art of Computer Programming ; Vol. 1. Fundamental Algorithms. Addison-Wesley, Reading 1968 (third edition : 1997)

Références citées :

Aho, A.V., Hopcroft, J.E., Ullman, J.D. [1974] : The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading 1974

Cobham, A. [1964] : The intrinsic computational difficulty of functions. Proceedings of the 1964 Congress for Logic Methodology and Philosophy of Science (Y. Bar-Hillel, ed.), North-Holland, Amsterdam 1964, pp. 24-30

Edmonds, J. [1965] : Paths, trees, and flowers. Canadian Journal of Mathematics 17 (1965), 449-467

Han, Y. [2004] : Deterministic sorting in $O(n \log \log n)$ time and linear space. Journal of Algorithms 50 (2004), 96-105

Stirling, J. [1730] : Methodus Differentialis, London 1730

Chapitre 2

Graphes

Les graphes seront utilisés tout au long de ce livre. Dans ce chapitre nous donnerons les définitions de base et nous préciserons nos notations. Nous présenterons également quelques théorèmes classiques et quelques algorithmes fondamentaux.

Après les définitions du paragraphe 2.1, nous étudierons quelques structures essentielles souvent rencontrées dans ce livre : les arbres, les cycles, les coupes. Nous démontrerons quelques propriétés importantes, et nous considérerons des systèmes d'ensembles reliés aux arbres au paragraphe 2.2. L'algorithme de recherche des composantes connexes ou fortement connexes sera présenté au paragraphe 2.3. Nous démontrerons le théorème d'Euler relatif aux parcours fermés qui passent une seule fois par chaque arête au paragraphe 2.4. Enfin, nous étudierons les graphes dessinables sur un plan sans que les arêtes se croisent aux paragraphes 2.5 et 2.6.

2.1 Définitions fondamentales

Un **graphe non orienté** est un triplet (V, E, Ψ) constitué de deux ensembles finis V et E et d'une application $\Psi : E \to \{X \subseteq V : |X| = 2\}$[1]. Un **graphe orienté** est un triplet (V, E, Ψ), constitué de deux ensembles finis V et E et d'une application $\Psi : E \to \{(v, w) \in V \times V : v \neq w\}$. V est l'ensemble des **sommets** du graphe ; E est l'ensemble de ses **arêtes** si le graphe est non orienté et de ses **arcs** s'il est orienté. Suivant un usage assez répandu, nous noterons également une arête $e = \{v, w\}$ par $e = (v, w)$ ou $e = (w, v)$.

Deux arêtes (arcs) e, e' seront dites **parallèles** si $\Psi(e) = \Psi(e')$. Un graphe sans arêtes ou arcs parallèles est un graphe **simple**. Quand un graphe est simple, nous pouvons identifier $e \in E$ avec son image $\Psi(e)$ et écrire $G = (V(G), E(G))$, avec $E(G) \subseteq \{X \subseteq V(G) : |X| = 2\}$ ou $E(G) \subseteq V(G) \times V(G)$. Nous utiliserons souvent cette notation même en présence d'arêtes (d'arcs) parallèles. Ainsi l'ensemble $E(G)$ pourra contenir plusieurs éléments «identiques». $|E(G)|$

[1] Nous utiliserons tout au long de cet ouvrage les notations ensemblistes anglophones (*ndt*).

est le nombre d'arêtes (arcs) ; si E et F sont deux ensembles d'arêtes (arcs), $|E \dot\cup F| = |E| + |F|$ même si des arêtes (arcs) parallèles apparaissent dans cette union.

Nous dirons qu'une arête (resp. un arc) $e = (v, w)$ **joint** v et w (resp. v à w) et que v et w sont **adjacents** ou mutuellement **voisins**. v et w seront les **extrémités** de e et nous dirons que v (resp. e) est **incident** à e (resp. à v). si $e = (v, w)$ est un arc, v est l'**origine** de e, w est l'**extrémité terminale** de e ; nous dirons que e est **sortant de** v et **entrant dans** w. Nous dirons aussi que v (resp. w) est le **voisin entrant** (resp. **voisin sortant**) de w (resp. v). Deux arcs ou arêtes ayant une extrémité commune seront dits **adjacents**.

La terminologie de la théorie des graphes n'est pas complètement figée. Par exemple, les sommets sont parfois appelés nœuds ou points ; en anglais, *edge* signifie arête ou arc. Un graphe avec des arêtes ou arcs parallèles est parfois appelé multigraphe. On peut également autoriser les boucles (extrémités identiques).

Si G est un graphe orienté, nous considérerons parfois son **graphe non orienté associé** G' obtenu en enlevant l'orientation de chaque arc de G. Nous dirons alors que G est une **orientation** de G'.

Un **sous-graphe** de $G = (V(G), E(G))$ est un graphe $H = (V(H), E(H))$ avec $V(H) \subseteq V(G)$ et $E(H) \subseteq E(G)$. Nous dirons que G contient H. Le graphe H est un **sous-graphe induit** de G si H est un sous-graphe de G et si $E(H) = \{(x, y) \in E(G) : x, y \in V(H)\}$; $H = G[V(H)]$ est le sous-graphe de G **induit par** $V(H)$. Un sous-graphe H de G est appelé **couvrant** si $V(H) = V(G)$.

Si $v \in V(G)$, $G - v$ est le sous-graphe de G induit par $V(G) \setminus \{v\}$. Si $e \in E(G)$, $G - e := (V(G), E(G) \setminus \{e\})$ est le graphe obtenu en supprimant e de E. $G + e := (V(G), E(G) \dot\cup \{e\})$ est le graphe obtenu en ajoutant une nouvelle arête (un nouvel arc) e à E. Si G et H sont deux graphes, $G + H$ est le graphe tel que $V(G + H) = V(G) \cup V(H)$ et tel que $E(G + H)$ est l'union disjointe de $E(G)$ et $E(H)$.

Deux graphes G et H sont appelés **isomorphes** s'il existe deux bijections $\Phi_V : V(G) \to V(H)$ et $\Phi_E : E(G) \to E(H)$ telles que $\Phi_E((v, w)) = (\Phi_V(v), \Phi_V(w))$ pour tout $(v, w) \in E(G)$. Nous ne distinguerons pas deux graphes isomorphes ; ainsi nous dirons que G contient H si G a un sous-graphe isomorphe à H.

Soit G un graphe non orienté et soit $X \subseteq V(G)$. Le graphe résultant de la **contraction** de X, et noté G/X, s'obtient en supprimant X et les arêtes de $G[X]$, puis en ajoutant un nouveau sommet x et en remplaçant enfin chaque arête (v, w) avec $v \in X$, $w \notin X$ par une arête (x, w) (notons que cette construction pourra créer des arêtes parallèles). Cette définition s'étend naturellement aux graphes orientés.

Soit un graphe G et $X, Y \subseteq V(G)$. Nous poserons : $E(X, Y) := \{(x, y) \in E(G) : x \in X \setminus Y,\ y \in Y \setminus X\}$ si G est non orienté et $E^+(X, Y) := \{(x, y) \in E(G) : x \in X \setminus Y,\ y \in Y \setminus X\}$ si G est orienté. Si G est non orienté et $X \subseteq V(G)$ nous poserons $\delta(X) := E(X, V(G) \setminus X)$. L'**ensemble des voisins** de X est défini par $\Gamma(X) := \{v \in V(G) \setminus X : E(X, \{v\}) \neq \emptyset\}$. Si G est orienté et $X \subseteq V(G)$ nous poserons : $\delta^+(X) := E^+(X, V(G) \setminus X)$, $\delta^-(X) := \delta^+(V(G) \setminus X)$ et $\delta(X) :=$

$\delta^+(X) \cup \delta^-(X)$. Nous utiliserons des indices (par exemple $\delta_G(X)$) pour spécifier le graphe G, si nécessaire.

Pour les ensembles de sommets ayant un seul élément $\{v\}$ ($v \in V(G)$), que nous appellerons aussi **singletons**, nous écrirons $\delta(v) := \delta(\{v\})$, $\Gamma(v) := \Gamma(\{v\})$, $\delta^+(v) := \delta^+(\{v\})$ et $\delta^-(v) := \delta^-(\{v\})$. Le **degré** d'un sommet v est $|\delta(v)|$, nombre d'arêtes incidentes à v. Dans le cas orienté, le **degré entrant** est $|\delta^-(v)|$, le **degré sortant** est $|\delta^+(v)|$, et le degré est $|\delta^+(v)|+|\delta^-(v)|$. Un sommet v de degré 0 est appelé **isolé**. Un graphe dont tous les sommets ont degré k est appelé ***k*-régulier**.

Si G est quelconque, $\sum_{v\in V(G)} |\delta(v)| = 2|E(G)|$. En particulier le nombre de sommets de G de degré impair est pair. Si G est orienté, $\sum_{v\in V(G)} |\delta^+(v)| = \sum_{v\in V(G)} |\delta^-(v)|$. Pour montrer ces relations, observons que chaque arc ou arête est compté deux fois dans chacun des membres de la première équation et que chaque arc est compté une fois dans chacun des membres de la deuxième équation. On peut aussi démontrer :

Lemme 2.1. *Soit G un graphe orienté et soient $X, Y \subseteq V(G)$:*

(a) $|\delta^+(X)|+|\delta^+(Y)| = |\delta^+(X\cap Y)|+|\delta^+(X\cup Y)|+|E^+(X,Y)|+|E^+(Y,X)|$;

(b) $|\delta^-(X)|+|\delta^-(Y)| = |\delta^-(X\cap Y)|+|\delta^-(X\cup Y)|+|E^+(X,Y)|+|E^+(Y,X)|$.

Soit G est un graphe non orienté et soient $X, Y \subseteq V(G)$:

(c) $|\delta(X)| + |\delta(Y)| = |\delta(X \cap Y)| + |\delta(X \cup Y)| + 2|E(X,Y)|$;

(d) $|\Gamma(X)| + |\Gamma(Y)| \geq |\Gamma(X \cap Y)| + |\Gamma(X \cup Y)|$.

Preuve. Il suffit d'utiliser des arguments de comptage. Soit $Z := V(G)\setminus(X\cup Y)$. Pour (a), observons que $|\delta^+(X)| + |\delta^+(Y)| = |E^+(X,Z)| + |E^+(X,Y \setminus X)| + |E^+(Y,Z)| + |E^+(Y,X \setminus Y)| = |E^+(X\cup Y,Z)| + |E^+(X\cap Y,Z)| + |E^+(X,Y \setminus X)| + |E^+(Y,X \setminus Y)| = |\delta^+(X\cup Y)| + |\delta^+(X\cap Y)| + |E^+(X,Y)| + |E^+(Y,X)|$. (b) se déduit de (a) en inversant l'orientation de chaque arc (remplacer (v,w) par (w,v)). (c) se déduit de (a) en remplaçant chaque arête (v,w) par une paire d'arcs de directions opposées (v,w) et (w,v).

Pour montrer (d), observons que $|\Gamma(X)| + |\Gamma(Y)| = |\Gamma(X \cup Y)| + |\Gamma(X) \cap \Gamma(Y)| + |\Gamma(X) \cap Y| + |\Gamma(Y) \cap X| \geq |\Gamma(X \cup Y)| + |\Gamma(X \cap Y)|$. □

Une fonction $f : 2^U \to \mathbb{R}$ (où U est un ensemble fini et 2^U est l'ensemble des parties de U) est appelée :

- **sous-modulaire** si $f(X\cap Y)+f(X\cup Y) \leq f(X)+f(Y)$ pour tout $X, Y \subseteq U$;
- **supermodulaire** si $f(X\cap Y)+f(X\cup Y) \geq f(X)+f(Y)$ pour tout $X, Y \subseteq U$;
- **modulaire** si $f(X \cap Y) + f(X \cup Y) = f(X) + f(Y)$ pour tout $X, Y \subseteq U$.

Le lemme 2.1 implique que $|\delta^+|$, $|\delta^-|$, $|\delta|$ et $|\Gamma|$ sont sous-modulaires. Cela sera utile ultérieurement.

Un **graphe complet** est un graphe simple non orienté tel que toute paire de sommets est adjacente. Le graphe complet à n sommets sera noté K_n. Le **complément** d'un graphe simple non orienté G est le graphe H tel que $G + H$ est un graphe complet.

Un **couplage** d'un graphe non orienté G est un ensemble d'arêtes deux à deux non adjacentes (c.-à-d. ayant toutes leurs extrémités différentes). Une **couverture par les sommets** de G est un ensemble $S \subseteq V(G)$ tel que chaque arête de G soit incidente à au moins un sommet dans S. Une **couverture par les arêtes** de G est un ensemble $F \subseteq E(G)$ d'arêtes tel que chaque sommet de G soit incident à au moins une arête de F. Un **ensemble stable** dans G est un ensemble de sommets deux à deux non adjacents. Un graphe sans aucune arête (ou arc) est dit **vide**. Une **clique** est un ensemble de sommets deux à deux adjacents.

Proposition 2.2. *Soit un graphe G et $X \subseteq V(G)$. Les propositions suivantes sont équivalentes :*

(a) *X est une couverture par les sommets dans G.*

(b) *$V(G) \setminus X$ est un ensemble stable dans G.*

(c) *$V(G) \setminus X$ est une clique dans le complément de G.* □

Si $\mathcal{F}$ est une famille d'ensembles ou de graphes, nous dirons que F est un élément **minimal** de $\mathcal{F}$ si $\mathcal{F}$ contient F, mais aucun sous-ensemble/sous-graphe propre de F. De même, F est **maximal** dans $\mathcal{F}$ si $F \in \mathcal{F}$ et F n'est pas un sous-ensemble/sous-graphe propre d'un élément de $\mathcal{F}$. Un élément **minimum** ou **maximum** est un élément de cardinalité minimum/maximum.

Une couverture par les sommets minimale n'est pas forcément minimum (voir par exemple figure 13.1), et un couplage maximal n'est en général pas maximum. Les problèmes de la recherche d'un couplage, d'un ensemble stable ou d'une clique maximum, de la couverture par les sommets ou par les arêtes minimum dans un graphe non orienté auront une grande importance dans la suite de ce livre.

Le **line graph**[2] d'un graphe simple non orienté G est le graphe $(E(G), F)$, tel que $F = \{(e_1, e_2) : e_1, e_2 \in E(G), |e_1 \cap e_2| = 1\}$. Notons que les couplages du graphe G correspondent aux ensembles stables du *line graph* de G.

Soit G un graphe orienté ou non. Une suite $P = [v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}]$ est un **parcours de v_1 à v_{k+1}** de G si $k \geq 0$ et si les deux **extrémités** de e_i sont v_i et v_{i+1} pour $i = 1, \ldots, k$. v_1 et v_{k+1} sont les **extrémités** de P. Si G est orienté, nous dirons que P est un **parcours orienté** quand l'arc e_i est sortant de v_i et entrant dans v_{i+1} pour $i = 1, \ldots, k$. v_1 sera l'**origine** du parcours P et v_{k+1} sera son **extrémité**. Si $e_i \neq e_j$ pour $1 \leq i < j \leq k$, P est un **parcours simple** de G. Un parcours simple P est **fermé** si $v_1 = v_{k+1}$. Un parcours simple P tel que $v_i \neq v_j$ pour $1 \leq i < j \leq k+1$ est une **chaîne**. Dans ce cas on pourra identifier P avec le sous-graphe $(\{v_1, \ldots, v_{k+1}\}, \{e_1, \ldots, e_k\})$ de G. Si P est une chaîne et $x, y \in V(P)$ $P_{[x,y]}$ sera l'(unique) sous-graphe de P qui est une chaîne de x à y. Il est évident qu'il existe un parcours d'un sommet v à un sommet $w \neq v$ si et seulement s'il existe une chaîne de v à w.

[2] En français «graphe représentatif des arêtes d'un graphe» ; nous garderons ici l'expression anglaise, plus concise et plus facile à utiliser (*ndt*).

Un parcours orienté qui est également une chaîne est un **chemin**. Deux sommets d'un graphe non orienté sont **connectés** s'il existe une chaîne ayant ces deux sommets comme extrémités ; un sommet t est **connecté** à un sommet s dans un graphe orienté s'il existe un chemin d'origine s et d'extrémité t.

Un graphe $(\{v_1, \ldots, v_k\}, \{e_1, \ldots, e_k\})$ tel que $v_1, e_1, v_2, \ldots, v_k, e_k, v_1$ est un parcours (resp. un parcours orienté) avec $v_i \neq v_j$ si $1 \leq i < j \leq k$ est un **cycle** (resp. un **circuit**). Un argument simple d'induction montre que l'ensemble des arêtes (resp. des arcs) d'un parcours fermé (resp. d'un parcours orienté fermé) peut être partitionné en ensembles d'arêtes de cycles (resp. d'arcs de circuits).

La **longueur** d'une chaîne (resp. d'un chemin) est son nombre d'arêtes (resp. d'arcs). Une chaîne qui couvre les sommets d'un graphe non orienté G est appelée **chaîne hamiltonienne** de G ; un cycle couvrant les sommets de G est appelé **cycle hamiltonien** ou **tour** de G. Un graphe contenant un cycle hamiltonien est appelé **graphe hamiltonien**. Les chemins et circuits hamiltoniens se définissent de la même manière quand les graphes sont orientés.

Si v et w sont deux sommets de G, la **distance** de v à w notée $\mathrm{dist}(v, w)$ ou $\mathrm{dist}_G(v, w)$ est la longueur d'un plus court chemin de v à w si G est orienté, et d'une plus courte chaîne de v à w si G est non orienté. S'il n'existe aucune chaîne (ou chemin dans le cas orienté) de v à w, nous poserons $\mathrm{dist}(v, w) := \infty$. Si G est non orienté, on a toujours $\mathrm{dist}(v, w) = \mathrm{dist}(w, v)$ pour toute paire $v, w \in V(G)$.

Souvent une fonction coût $c : E(G) \to \mathbb{R}$ sera associée aux problèmes que nous étudierons. Si $F \subseteq E(G)$, nous poserons $c(F) := \sum_{e \in F} c(e)$ (et $c(\emptyset) = 0$). La fonction $c : 2^{E(G)} \to \mathbb{R}$ est une fonction modulaire. $\mathrm{dist}_{(G,c)}(v, w)$ sera le minimum de $c(E(P))$ pour toutes les chaînes (chemins) P de v à w.

2.2 Arbres, cycles, coupes

Un graphe orienté ou non orienté G sera dit **connexe** s'il existe une chaîne de v à w pour tous $v, w \in V(G)$; sinon G sera **non connexe**. Les sous-graphes connexes maximaux de G sont les **composantes connexes** de G. Nous identifierons quelquefois les composantes connexes avec les ensembles de sommets qu'elles induisent. Un ensemble de sommets X est connexe si le sous-graphe induit par X est connexe. v est un **sommet d'articulation** si $G - v$ a plus de composantes connexes que G ; $e \in E(G)$ est un **pont** si $G - e$ a plus de composantes connexes que G.

Un graphe non orienté sans cycles est une **forêt**. Une forêt connexe est un **arbre**. Dans un arbre, une **feuille** est un sommet de degré 1. Une **étoile** est un arbre ayant au plus un sommet qui n'est pas une feuille.

Nous allons maintenant donner quelques propriétés des arbres et des arborescences, leurs analogues dans les graphes orientés. Établissons le résultat suivant :

Proposition 2.3.

(a) *Un graphe orienté ou non orienté G est connexe si et seulement si $\delta(X) \neq \emptyset$ pour tout $\emptyset \neq X \subset V(G)$.*

(b) *Soit G un graphe orienté et soit $r \in V(G)$. Il existe un chemin de r à tout sommet $v \in V(G)$ si et seulement si $\delta^+(X) \neq \emptyset$ pour tout $X \subset V(G)$ contenant r.*

Preuve. (a) : s'il existe $X \subset V(G)$ tel que $r \in X$, $v \in V(G) \setminus X$, et $\delta(X) = \emptyset$, il ne peut exister de chaîne de r à v et G n'est pas connexe. Inversement, si G n'est pas connexe, il existe deux sommets r, v non connectés. Si R est l'ensemble des sommets connectés à r, $r \in R$, $v \notin R$ et $\delta(R) = \emptyset$.

(b) : la preuve est analogue. □

Théorème 2.4. *Soit G un graphe non orienté connexe ayant n sommets. Les propositions suivantes sont équivalentes :*

(a) *G est un arbre (G est connexe et sans cycles).*

(b) *G est sans cycles et a $n-1$ arêtes.*

(c) *G est connexe et a $n-1$ arêtes.*

(d) *G est un graphe connexe minimal (chaque arête est un pont).*

(e) *G est un graphe minimal vérifiant la propriété $\delta(X) \neq \emptyset$ pour tout $\emptyset \neq X \subset V(G)$.*

(f) *G est un graphe maximal sans cycles (l'addition d'une arête quelconque crée un cycle).*

(g) *Toute paire de sommets de G est connectée par une chaîne unique.*

Preuve. (a)⇒(g), car l'union de deux chaînes distinctes ayant les mêmes extrémités contient un cycle.

(g)⇒(e)⇒(d) se déduit de la proposition 2.3(a).

(d)⇒(f) : évident.

(f)⇒(b)⇒(c) : cela se déduit du fait que si une forêt a n sommets, m arêtes et p composantes connexes, alors $n = m + p$. (Preuve par induction sur m.)

(c)⇒(a) : soit G connexe ayant $n-1$ arêtes. Si G a un cycle, on peut le supprimer en en retirant une arête. Supposons qu'après avoir retiré k arêtes de cette manière, le graphe résultant G' soit connexe et sans cycles. G' a $m = n-1-k$ arêtes. Comme $n = m + p = n - 1 - k + 1$, on a $k = 0$. □

En particulier, (d)⇒(a) signifie qu'un graphe est connexe si et seulement s'il contient un **arbre couvrant** (un sous-graphe couvrant qui est un arbre).

Un graphe orienté est une **ramification** si le graphe non orienté associé est une forêt et si chaque sommet v a au plus un arc entrant. Une ramification connexe est une **arborescence**. Par le théorème 2.4, une arborescence ayant n sommets a $n-1$ arcs et par conséquent il n'existe qu'un sommet r avec $\delta^-(r) = \emptyset$. Ce sommet est appelé la **racine** de l'arborescence ; nous dirons aussi que l'arborescence est **enracinée en** r. Les **feuilles** sont les sommets v qui vérifient $\delta^+(v) = \emptyset$.

Théorème 2.5. *Soit G un graphe orienté ayant n sommets. Les propositions suivantes sont équivalentes :*

(a) *G est une arborescence enracinée en r (c.-à-d. une ramification connexe avec $\delta^-(r) = \emptyset$).*

(b) *G est une ramification avec $n-1$ arcs et $\delta^-(r) = \emptyset$.*

(c) *G a $n-1$ arcs et chaque sommet est connecté à r.*

(d) *Chaque sommet est connecté à r, mais cette propriété n'est plus vraie si on supprime un arc quelconque.*

(e) *G est minimal pour la propriété : si $X \subset V(G)$ et $r \in X$, alors $\delta^+(X) \neq \emptyset$.*

(f) *$\delta^-(r) = \emptyset$, il existe un seul parcours orienté de r à v pour tout $v \in V(G) \setminus \{r\}$.*

(g) *$\delta^-(r) = \emptyset$, $|\delta^-(v)| = 1$ si $v \in V(G) \setminus \{r\}$ et G ne contient pas de circuits.*

Preuve. (a)⇒(b) et (c)⇒(d) se déduisent du théorème 2.4.

(b)⇒(c) : comme $|\delta^-(v)| = 1$ si $v \in V(G) \setminus \{r\}$, il existe pour tout v un chemin de r à v (partir de v et toujours utiliser l'arc entrant jusqu'à atteindre r).

(d)⇒(e) se déduit de la proposition 2.3(b).

(e)⇒(f) : $\delta^-(r) = \emptyset$ par la minimalité de (e). Par la proposition 2.3(b), il existe un chemin de r à v pour tout v. S'il existe deux parcours orientés de r à v, le dernier arc d'un de ces parcours qui n'est pas dans l'autre est tel que chaque sommet de G continue à être connecté à r après suppression de cet arc, ce qui contredit (e) par la proposition 2.3(b).

(f)⇒(g)⇒(a) : trivial. □

Une **coupe** dans un graphe G est un ensemble d'arêtes $\delta(X)$ avec $\emptyset \neq X \subset V(G)$. Nous dirons que $\delta(X)$ sépare deux sommets s et t si $s \in X$ et $t \notin X$. Si G est orienté, $\delta^+(X)$ est une coupe **orientée** si $\emptyset \neq X \subset V(G)$ et $\delta^-(X) = \emptyset$, c.-à-d. si aucun arc n'est entrant dans X. Un ensemble d'arcs $\delta^+(X)$ tel que $s \in X$ et $t \notin X$ est une **coupe séparant t de s**. Un ensemble d'arcs $\delta^+(X)$ tel que $\emptyset \neq X \subset V(G)$ sera appelé **coupe sortante** ; si, de plus $r \in X$, nous dirons que $\delta^+(X)$ est une **coupe issue de r**.

Lemme 2.6. (Minty [1960]) *Soit G un graphe orienté et soit $e \in E(G)$. Supposons que e soit colorié en noir, les autres arcs étant coloriés en noir, rouge ou vert. Une et une seule des propositions suivantes est vraie :*

(a) *Il existe un cycle contenant e et colorié en rouge et noir de telle sorte que les arcs noirs aient la même orientation.*

(b) *Il existe une coupe contenant e et coloriée en vert et noir de telle sorte que les arcs noirs aient la même orientation.*

Preuve. Soit $e = (x, y)$. Attribuons des labels aux sommets de G par la procédure suivante : donnons d'abord un label à y. Si v a un label et w n'en a pas, donnons à w le label $pred(w) := v$ quand il existe un arc noir (v, w), un arc rouge (v, w) ou un arc rouge (w, v).

À la fin de cette procédure, il existe deux possibilités :
Cas 1 : x a reçu un label. Alors les sommets $x, pred(x), pred(pred(x)), \ldots, y$ induisent un cycle vérifiant la propriété (a).
Cas 2 : x n'a pas reçu de label. Soit R l'ensemble des sommets ayant un label. La coupe $\delta^+(R) \cup \delta^-(R)$ vérifie alors la propriété (b).

Supposons qu'il existe simultanément un cycle C vérifiant (a) et une coupe $\delta^+(X) \cup \delta^-(X)$ vérifiant (b). Tous les arcs de l'intersection sont noirs et ils ont tous la même orientation sur C ; mais ils sont aussi tous entrant dans X ou sortant de X, ce qui est une contradiction. □

Un graphe orienté est appelé **fortement connexe** s'il existe un chemin de s à t et un chemin de t à s pour toute paire $s, t \in V(G)$. Les **composantes fortement connexes** d'un graphe orienté sont les sous-graphes maximaux fortement connexes.

Corollaire 2.7. *Dans un graphe orienté G, tout arc appartient soit à un circuit, soit à une coupe orientée. De plus, les propositions suivantes sont équivalentes :*

(a) *G est fortement connexe.*

(b) *G ne contient aucune coupe orientée.*

(c) *G est connexe et chaque arc de G appartient à un circuit.*

Preuve. En coloriant tous les arcs en noir dans le lemme de Minty, on montre facilement la première condition de l'énoncé ainsi que (b)⇒(c).

(a)⇒(b) est une conséquence de la proposition 2.3(b).

(c)⇒(a) : soit $r \in V(G)$ un sommet arbitraire. S'il n'existe pas de chemin de r à v pour $v \in V(G)$, il existe par la proposition 2.3(b) $X \subset V(G)$ avec $r \in X$ tel que $\delta^+(X) = \emptyset$. Comme G est connexe, $\delta^+(X) \cup \delta^-(X) \neq \emptyset$ (par la proposition 2.3(a)) ; soit $e \in \delta^-(X)$. e ne peut appartenir à aucun circuit puisque aucun arc n'est sortant de X. □

Le corollaire 2.7 et le théorème 2.5 impliquent qu'un graphe orienté est fortement connexe si et seulement s'il existe pour chaque sommet v une arborescence couvrante enracinée en v.

Par le corollaire 2.7, un graphe orienté est **sans circuit** si et seulement si chaque arc appartient à une coupe orientée. De plus, un graphe orienté est sans circuit si et seulement si les composantes fortement connexes sont des singletons. Les sommets d'un graphe orienté sans circuit peuvent être ordonnés de la manière simple :

Définition 2.8. *Soit G un graphe orienté. Un* **ordre topologique** *de G est un ordre des sommets $V(G) = \{v_1, \ldots, v_n\}$ tel que $(v_i, v_j) \in E(G)$, implique $i < j$.*

Proposition 2.9. *Un graphe orienté a un ordre topologique si et seulement s'il est sans circuit.*

Preuve. Si un graphe orienté a un circuit, il ne peut manifestement pas exister d'ordre topologique. Montrons la réciproque par induction sur le nombre d'arcs. S'il n'existe aucun arc, tout ordre est topologique. Sinon soit $e \in E(G)$; par le corollaire

2.7 e appartient à une coupe orientée $\delta^+(X)$. Mais alors un ordre topologique de $G[X]$ suivi d'un ordre topologique de $G - X$ (les deux existent par l'hypothèse d'induction) définit un ordre topologique de G. □

Les cycles et les coupes jouent un rôle important dans la théorie algébrique des graphes. Associons à un graphe G un espace vectoriel $\mathbb{R}^{E(G)}$ dont les éléments sont les vecteurs $(x_e)_{e\in E(G)}$ avec $|E(G)|$ composantes à valeur réelle. En nous référant à Berge [1985], nous allons brièvement étudier deux sous-espaces linéaires particulièrement importants.

Soit G un graphe orienté. À chaque cycle C de G, associons un vecteur $\zeta(C) \in \{-1, 0, 1\}^{E(G)}$ défini par $\zeta(C)_e = 0$ si $e \notin E(C)$ et par $\zeta(C)_e \in \{-1, 1\}$ si $e \in E(C)$ de telle sorte qu'en inversant l'orientation de tous les arcs e tels que $\zeta(C)_e = -1$ nous obtenions un circuit. De même, associons à toute coupe $D = \delta(X)$ de G un vecteur $\zeta(D) \in \{-1, 0, 1\}^{E(G)}$ défini par $\zeta(D)_e = 0$ si $e \notin D$, et par $\zeta(D)_e = -1$ si $e \in \delta^-(X)$ et $\zeta(D)_e = 1$ si $e \in \delta^+(X)$. Ces vecteurs sont définis à une multiplication par -1 près. Donc les sous-espaces vectoriels de $\mathbb{R}^{E(G)}$ engendrés par les vecteurs associés aux cycles de G d'une part et par les vecteurs associés aux coupes de G d'autre part sont bien définis ; ils définissent ce que nous nommerons respectivement l'**espace des cycles** et l'**espace des cocycles** de G.

Proposition 2.10. *L'espace des cycles est orthogonal à l'espace des cocycles.*

Preuve. Soit C un cycle et soit $D = \delta(X)$ une coupe. Montrons que le produit scalaire de $\zeta(C)$ est de $\zeta(D)$ est égal à zéro : puisque la réorientation des arcs ne change pas le produit scalaire, nous pouvons supposer que D est une coupe orientée. Il suffit alors d'observer que, quand on décrit un cycle, on entre autant de fois dans X qu'on en sort. □

Nous allons montrer maintenant que la somme des dimensions de l'espace des cycles et de l'espace des cocycles est $|E(G)|$, dimension de l'espace vectoriel. Un ensemble de cycles (de coupes) est une **base des cycles** (une **base des cocycles**) si les vecteurs associés forment une base de l'espace des cycles (respectivement, de l'espace des cocycles). Soit T un sous-graphe maximal sans cycle de G. Pour chaque $e \in E(G) \setminus E(T)$ l'unique cycle dans $T + e$ sera le **cycle fondamental** de e par rapport à T. De plus, il existe pour chaque $e \in E(T)$ un ensemble $X \subseteq V(G)$ tel que $\delta_G(X) \cap E(T) = \{e\}$ (prenons une des composantes connexes de $T - e$) ; $\delta_G(X)$ sera la **coupe fondamentale** de e par rapport à T.

Théorème 2.11. *Soit G un graphe orienté et soit T un sous-graphe maximal sans circuit. Les $|E(G) \setminus E(T)|$ cycles fondamentaux par rapport à T forment une base des cycles de G, et les $|E(T)|$ coupes fondamentales par rapport à T forment une base des cocycles de G.*

Preuve. Les vecteurs associés aux cycles fondamentaux sont linéairement indépendants puisque chaque cycle fondamental contient un élément qui n'appartient à aucun autre. Cela est également vrai pour les coupes fondamentales. Comme les deux

sous-espaces sont orthogonaux entre eux par la proposition 2.10, la somme de leur dimension ne peut excéder $|E(G)| = |E(G) \setminus E(T)| + |E(T)|$. □

Les coupes fondamentales ont des propriétés importantes que nous allons souvent exploiter et que nous allons présenter maintenant. Soit T un graphe orienté dont le graphe non orienté associé est un arbre. Considérons la famille $\mathcal{F} := \{C_e : e \in E(T)\}$, où C_e est la composante connexe de $T - e$ contenant y si $e = (x, y) \in E(T)$; $\delta(C_e)$ est donc la coupe fondamentale de e par rapport à T. Si T est une arborescence, deux éléments quelconques de $\mathcal{F}$ sont deux ensembles disjoints ou alors un des deux est un sous-ensemble de l'autre. En général $\mathcal{F}$ est sans croisements.

Définition 2.12. *Un* **système d'ensembles** *est une paire $(U, \mathcal{F})$, où U est un ensemble fini non vide et $\mathcal{F}$ est une famille de sous-ensembles de U. $(U, \mathcal{F})$ est* **sans croisements** *si pour toute paire $X, Y \in \mathcal{F}$, au moins un des quatre ensembles $X \setminus Y$, $Y \setminus X$, $X \cap Y$, $U \setminus (X \cup Y)$ est vide. $(U, \mathcal{F})$ est* **laminaire** *si pour toute paire $X, Y \in \mathcal{F}$, au moins un des trois ensembles $X \setminus Y$, $Y \setminus X$, $X \cap Y$ est vide.*

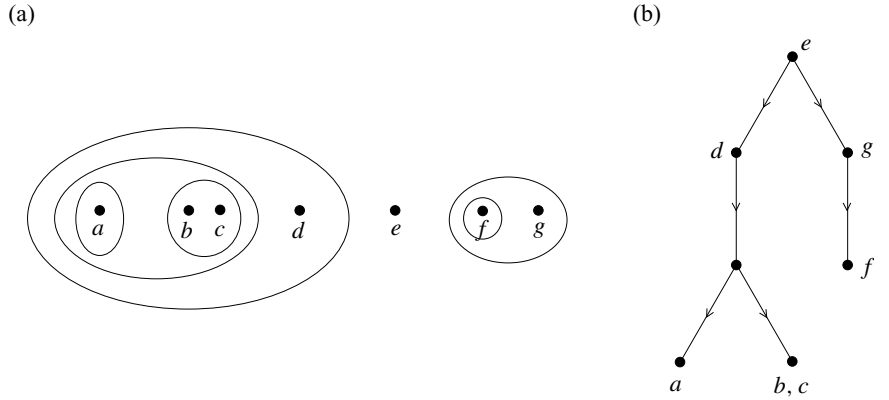


Figure 2.1.

Dans la littérature, les systèmes d'ensembles sont connus sous le nom d'**hypergraphes**. On se reportera à la figure 2.1(a) pour une représentation de la famille laminaire $\{\{a\}, \{b, c\}, \{a, b, c\}, \{a, b, c, d\}, \{f\}, \{f, g\}\}$.

Comme la propriété d'être laminaire pour un système d'ensembles $(U, \mathcal{F})$ ne dépend pas de U, nous dirons parfois que $\mathcal{F}$ est une famille laminaire. Cela n'est cependant pas exact pour les systèmes d'ensembles sans croisements. Ainsi, si U contient un élément qui n'appartient à aucun ensemble de $\mathcal{F}$, alors $\mathcal{F}$ est sans croisements si et seulement si $\mathcal{F}$ est laminaire. Choisissons un élément arbitraire $r \in U$. Il découle immédiatement des définitions qu'un système d'ensembles $(U, \mathcal{F})$ est sans croisements si et seulement si

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{U \setminus X : X \in \mathcal{F},\, r \in X\}$$

est laminaire. Ainsi, les familles sans croisements seront parfois décrites comme le sont les familles laminaires : par exemple, la figure 2.2(a) décrit la famille sans croisements $\{\{b,c,d,e,f\},\{c\},\{a,b,c\},\{e\},\{a,b,c,d,f\},\{e,f\}\}$; un carré correspond à l'ensemble contenant tous les élements à l'extérieur de ce carré.

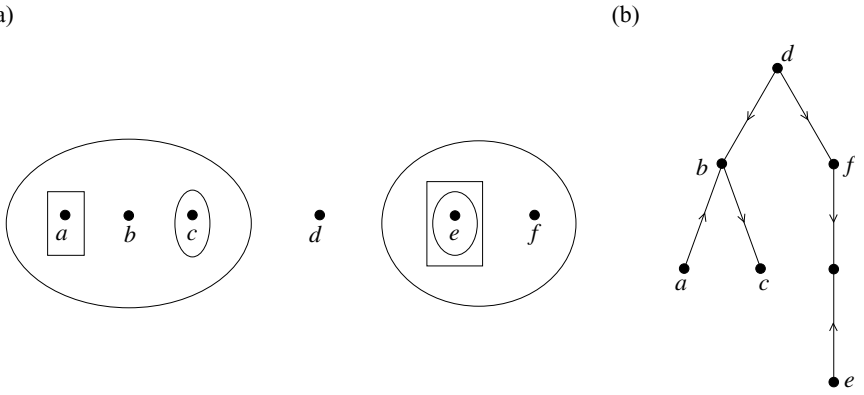


Figure 2.2.

Nous avons vu que les arbres orientés décrivent les familles sans croisements ; le contraire est également vrai. Toute famille sans croisements peut être décrite par un arbre dans la représentation suivante :

Définition 2.13. *Soit T un graphe orienté obtenu par orientation d'un arbre. Soit U un ensemble fini et soit $\varphi : U \to V(T)$. Soit $\mathcal{F} := \{S_e : e \in E(T)\}$; S_e étant ainsi associé à $e = (x, y)$:*

$$S_e := \{s \in U : \varphi(s) \text{ est dans la même composante connexe de } T - e \text{ que } y\}.$$

Alors (T, φ) est appelée **représentation par arbre** *de $(U, \mathcal{F})$.*

Voir les figures 2.1(b) et 2.2(b) comme exemples.

Proposition 2.14. *Soit $(U, \mathcal{F})$ un système d'ensembles avec une représentation par arbre (T, φ). Alors $(U, \mathcal{F})$ est sans croisements. Si T est une arborescence, alors $(U, \mathcal{F})$ est laminaire. De plus, toute famille sans croisements a une représentation par arbre et, si la famille est laminaire, on peut supposer que T est une arborescence.*

Preuve. Si (T, φ) est une représentation par arbre de $(U, \mathcal{F})$ et $e = (v, w), f = (x, y) \in E(T)$, il existe P, chaîne de v à x dans T. Il y a alors quatre cas : si $w, y \notin V(P)$ alors $S_e \cap S_f = \emptyset$ (puisque T n'a pas de cycles) ; si $w \notin V(P)$ et $y \in V(P)$ alors $S_e \subseteq S_f$; si $y \notin V(P)$ et $w \in V(P)$ alors $S_f \subseteq S_e$; si

$w, y \in V(P)$ alors $S_e \cup S_f = U$. Donc $(U, \mathcal{F})$ est sans croisements. Si T est une arborescence, le dernier cas ne peut pas se produire (sinon au moins un sommet de P aurait deux arcs entrants) et $\mathcal{F}$ est laminaire.

Pour montrer la réciproque, considérons d'abord une famille laminaire $\mathcal{F}$. Soit $V(T) := \mathcal{F} \dot\cup \{r\}$ et $E(T) :=$

$$\begin{aligned} & \{(X,Y) \in \mathcal{F} \times \mathcal{F} : X \supset Y \neq \emptyset \text{ et il n'y a aucun } Z \in \mathcal{F} \text{ avec } X \supset Z \supset Y\} \\ \cup \quad & \{(r,X) : X = \emptyset \in \mathcal{F} \text{ ou } X \text{ est un élément maximal de } \mathcal{F}\}. \end{aligned}$$

Nous poserons $\varphi(x) := X$, où X est l'ensemble minimal dans $\mathcal{F}$ contenant x, et $\varphi(x) := r$ si aucun ensemble $\mathcal{F}$ ne contient x. T est manifestement une arborescence enracinée en r, et (T, φ) est une représentation en arbre de $\mathcal{F}$.

Supposons maintenant que $\mathcal{F}$ soit une famille sans croisements de sous-ensembles de U. Soit $r \in U$; comme nous l'avons vu précédemment,

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{U \setminus X : X \in \mathcal{F},\, r \in X\}$$

est laminaire. Soit alors (T, φ) une représentation en arbre de $(U, \mathcal{F}')$. Si $e \in E(T)$ est un arc, il y a trois cas à étudier : si $S_e \in \mathcal{F}$ et $U \setminus S_e \in \mathcal{F}$, nous remplaçons $e = (x, y)$ par deux arcs (x, z) et (y, z), où z est un nouveau sommet. Si $S_e \notin \mathcal{F}$ et $U \setminus S_e \in \mathcal{F}$, nous remplaçons l'arc $e = (x, y)$ par (y, x). Si $S_e \in \mathcal{F}$ et $U \setminus S_e \notin \mathcal{F}$, nous ne faisons rien. Soit T' le graphe résultant. (T', φ) est une représentation en arbre de $(U, \mathcal{F})$. □

Le résultat précédent est mentionné par Edmonds et Giles [1977], mais était probablement déjà connu.

Corollaire 2.15. *Une famille laminaire de sous-ensembles distincts de U a au plus $2|U|$ éléments. Un famille sans croisements de sous-ensembles distincts de U a au plus $4|U| - 2$ éléments.*

Preuve. Considérons d'abord une famille laminaire $\mathcal{F}$ de sous-ensembles propres et distincts de U. Montrons que $|\mathcal{F}| \leq 2|U| - 2$. Soit (T, φ) une représentation en arbre, où T est une arborescence avec le plus petit nombre possible de sommets. Pour tout $w \in V(T)$, soit $|\delta^+(w)| \geq 2$, soit il existe $x \in U$ tel que $\varphi(x) = w$ ou alors ces deux conditions sont vraies. (Pour la racine cela se déduit de $U \notin \mathcal{F}$, pour les feuilles de $\emptyset \notin \mathcal{F}$, pour les autres sommets de la minimalité de T.)

Il y a au plus $|U|$ sommets w tels que $\varphi(x) = w$ pour un $x \in U$ et au plus $\left\lfloor \frac{|E(T)|}{2} \right\rfloor$ sommets w avec $|\delta^+(w)| \geq 2$. Donc $|E(T)| + 1 = |V(T)| \leq |U| + \frac{|E(T)|}{2}$ ce qui implique $|\mathcal{F}| = |E(T)| \leq 2|U| - 2$.

Considérons maintenant $(U, \mathcal{F})$ une famille sans-croisements avec $\emptyset, U \notin \mathcal{F}$, et soit $r \in U$. Puisque

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{U \setminus X : X \in \mathcal{F},\, r \in X\}$$

est laminaire, $|\mathcal{F}'| \leq 2|U| - 2$. Donc $|\mathcal{F}| \leq 2|\mathcal{F}'| \leq 4|U| - 4$. Nous concluons en remarquant que $\emptyset$ et U peuvent être des membres de $\mathcal{F}$. □

2.3 Connexité

La connexité est une notion très importante en théorie des graphes. Pour de nombreux problèmes, il suffit de supposer que les graphes sont connexes ; dans le cas contraire on pourra résoudre le problème étudié dans chaque composante connexe séparément. Il est donc essentiel de savoir trouver les composantes connexes d'un graphe. L'algorithme élémentaire suivant trouve un chemin (ou une chaîne dans le cas non orienté) entre un sommet donné s et tous les sommets connectés à s. Il s'applique aussi bien aux graphes orientés qu'aux graphes non orientés. Dans le cas non orienté il construit un arbre maximal contenant s ; dans le cas orienté il construit une arborescence maximale de racine s.

ALGORITHME DE BALAYAGE DE GRAPHES	
Input	Un graphe G (orienté ou non) et un sommet s.
Output	(R, T) qui est une arborescence de racine s, ou un arbre, R étant l'ensemble des sommets connectés à s et $T \subseteq E(G)$.

① $R := \{s\}$, $Q := \{s\}$ et $T := \emptyset$.

② **If** $Q = \emptyset$ **then stop**,
else choisir $v \in Q$.

③ Choisir $w \in V(G) \setminus R$ tel que $e = (v, w) \in E(G)$.
If aucun w n'existe **then** $Q := Q \setminus \{v\}$ et **go to** ②.

④ $R := R \cup \{w\}$, $Q := Q \cup \{w\}$ et $T := T \cup \{e\}$. **Go to** ②.

Proposition 2.16. *L'*ALGORITHME DE BALAYAGE DE GRAPHES *répond correctement.*

Preuve. À chaque étape, (R, T) est un arbre ou une arborescence enracinée en s. Supposons qu'à la fin de l'algorithme, il existe $w \in V(G) \setminus R$ connecté à s. Soit P un chemin de s à w (resp. une chaîne de s à w) et soit (x, y) un arc (resp. une arête) de P avec $x \in R$ et $y \notin R$. Puisque x appartient à R, il a aussi appartenu à Q au cours du déroulement de l'algorithme. L'algorithme ne s'arrête pas avant d'avoir effacé x de Q. Mais cela se fait seulement en ③ s'il n'existe aucun arc (resp. aucune arête) (x, y) avec $y \notin R$. □

Puisque cet algorithme de graphe est le premier que nous présentons dans ce livre, intéressons-nous à quelques problèmes d'implémentation. La première question concerne la manière de représenter le graphe ; plusieurs représentations naturelles existent. On peut par exemple imaginer une matrice dont les lignes sont indicées par les sommets et les colonnes par les arêtes (ou arcs). La **matrice d'incidence** d'un graphe non orienté G est la matrice $A = (a_{v,e})_{v \in V(G),\, e \in E(G)}$ où

$$a_{v,e} = \begin{cases} 1 & \text{si } v \in e \\ 0 & \text{si } v \notin e \end{cases}.$$

Si G est orienté, sa **matrice d'incidence** est la matrice $A = (a_{v,e})_{v \in V(G),\, e \in E(G)}$ où

$$a_{v,(x,y)} = \begin{cases} -1 & \text{si } v = x \\ 1 & \text{si } v = y \\ 0 & \text{si } v \notin \{x, y\} \end{cases} .$$

Cependant cette méthode n'est pas très performante puisque chaque colonne contient seulement deux éléments non nuls. L'espace nécessaire pour stocker une matrice d'incidence est $O(nm)$, où $n := |V(G)|$ et $m := |E(G)|$.

Une meilleure méthode consiste à construire la matrice dont les lignes et colonnes sont indicées par l'ensemble des sommets. La **matrice d'adjacence** d'un graphe simple G est la matrice binaire A dont les composantes $a_{v,w}$ valent 1 si et seulement si $(v, w) \in E(G)$. Pour les graphes ayant des arêtes ou des arcs parallèles, $a_{v,w}$ est le nombre d'arêtes ou d'arcs entre v et w. Une matrice d'adjacence nécessite un espace $O(n^2)$ dans le cas des graphes simples.

La matrice d'adjacence convient bien si le graphe est **dense**, c.-à-d. a $\Theta(n^2)$ arêtes ou arcs (ou plus). Pour les graphes **peu denses** avec, disons, seulement $O(n)$ arêtes ou arcs, on peut faire nettement mieux. Avant de stocker le nombre de sommets, commençons par fournir une liste des arêtes ou arcs, ainsi que leurs extrémités. Si chaque sommet est représenté par un entier entre 1 et n, l'espace nécessaire pour chaque arête ou arc est $O(\log n)$. Au total nous avons besoin d'un espace $O(m \log n)$.

Stocker les arêtes ou arcs dans un ordre quelconque n'est pas très judicieux. Dans presque tous les algorithmes de graphes il est nécessaire de trouver les arêtes ou arcs incidents à un sommet donné. Il faut donc avoir pour chaque sommet une liste des arcs ou arêtes qui lui sont incidents. Dans le cas orienté, il faudra deux listes, l'une pour les arcs entrants et l'autre pour les arcs sortants. Cette structure de données est appelée **liste d'adjacence** ; c'est la plus utile pour les graphes. Pour un accès direct à la ou aux deux listes, il nous faut des pointeurs sur les emplacements de tête de la ou des deux listes de chaque sommet. Cela demande un espace supplémentaire de $O(n \log m)$ bits. Le nombre total de bits nécessaires pour décrire une liste d'adjacence est donc $O(n \log m + m \log n)$.

Désormais, chaque fois qu'un graphe fera partie de l'input d'un algorithme, nous supposerons qu'il sera donné par sa liste d'adjacence.

Comme pour les opérations élémentaires sur les nombres (voir paragraphe 1.2), nous supposerons que les opérations élémentaires sur les sommets et arêtes se font en temps constant. Cela comprend l'accès à l'enregistrement d'une arête, l'identification de ses deux extrémités, l'accès à la tête de la liste d'adjacence pour un sommet. Le temps de calcul sera mesuré par les paramètres n et m, et un algorithme dont le temps de calcul est $O(m + n)$ est appelé linéaire.

Nous utiliserons toujours les lettres n et m pour désigner le nombre de sommets et d'arêtes ou arcs. Pour de nombreux algorithmes de graphes on peut supposer que le graphe que l'on étudie est connexe ; on a donc $n - 1 \leq m < n^2$. Souvent en cas de présence d'arêtes ou d'arcs parallèles on peut n'en garder qu'une ; d'autre part des composantes connexes distinctes peuvent être examinées séparément. Ces

opérations préparatoires peuvent s'implémenter à l'avance en temps linéaire ; voir l'exercice 13 et la suite de ce paragraphe.

Nous pouvons maintenant analyser la complexité de l'ALGORITHME DE BALAYAGE DE GRAPHES :

Proposition 2.17. *L'*ALGORITHME DE BALAYAGE DE GRAPHES *peut être implémenté en un temps $O(m+n)$. Les composantes connexes d'un graphe peuvent être trouvées en temps linéaire.*

Preuve. Supposons que G soit donné par sa liste d'adjacence. À tout sommet x associons un pointeur *courant*(x), indiquant l'arc de la liste $\delta^+(x)$ ou l'arête de la liste $\delta(x)$ (cette liste fait partie de l'input) examiné à l'itération courante. Initialement *courant*(x) est positionné sur le premier élément de la liste. Dans ③, le pointeur s'incrémente. Quand la fin de la liste est atteinte, x est retiré de Q et ne sera plus jamais réinséré. Donc, au total, le temps de calcul est proportionnel au nombre de sommets augmenté du nombre d'arêtes ou arcs c.-à-d. $O(n+m)$.

Pour identifier les composantes connexes d'un graphe, nous appliquons une seule fois l'algorithme et vérifions si $R = V(G)$. Si tel est le cas, le graphe est connexe. Sinon R est une composante connexe et nous appliquons l'algorithme à (G, s') pour un sommet arbitraire $s' \in V(G) \setminus R$ (et nous itérons jusqu'à épuisement de l'ensemble des sommets qui sont successivement ajoutés à R). De nouveau aucun arc ou arête n'est examiné deux fois et le temps total de calcul reste linéaire. □

Une question intéressante concerne l'ordre dans lequel les sommets sont choisis dans ③. On ne peut rien dire sur cet ordre si on ne précise pas la manière de choisir $v \in Q$ dans ②. Deux méthodes sont fréquemment utilisées ; la procédure DEPTH-FIRST SEARCH (DFS), en français **recherche en profondeur d'abord** et la procédure BREADTH-FIRST SEARCH (BFS), en français **recherche en largeur d'abord**. Dans (la procédure) DFS nous choisissons pour $v \in Q$ le dernier sommet entré dans Q. En d'autres termes, Q est implémenté suivant une pile LIFO (*last-in-first-out*, en français *dernier arrivé, premier sorti*). Dans BFS nous choisissons pour $v \in Q$ le premier sommet entré dans Q. Ici Q est implémenté suivant une pile FIFO (*first-in-first-out*, en français *premier arrivé, premier sorti*).

Un algorithme semblable à DFS a été proposé avant 1900 par Trémaux et Tarry ; voir König [1936]. Il semble que l'expression BFS ait été utilisée pour la première fois par Moore [1959]. Les arbres (dans le cas orienté : les arborescences) (R, T) calculés par DFS et BFS sont respectivement appelés **arbres-DFS** et **arbres-BFS**. Pour les arbres-BFS, il convient de noter l'importante propriété suivante :

Proposition 2.18. *Un arbre-*BFS *contient un plus court chemin (resp. une plus courte chaîne) de s à tous les sommets $v \in V(G)$ connectés à s. Les valeurs $\mathrm{dist}_G(s, v)$ se calculent en temps linéaire.*

Preuve. Nous appliquons BFS à (G, s) et ajoutons deux instructions : initialement (en ① de l'ALGORITHME DE BALAYAGE DE GRAPHES) nous posons $l(s) := 0$, et

en ④ nous posons $l(w) := l(v) + 1$. Il est évident que $l(v) = \mathrm{dist}_{(R,T)}(s, v)$ pour tout $v \in R$ à chaque étape de l'algorithme. De plus, si v est le sommet que l'algorithme est en train d'examiner (choisi en ②), il n'y a, à ce moment, aucun sommet $w \in R$ tel que $l(w) > l(v) + 1$ (parce que les sommets sont examinés par valeurs de l non décroissantes). Supposons qu'à la fin de l'algorithme, il existe un sommet $w \in V(G)$ avec $\mathrm{dist}_G(s, w) < \mathrm{dist}_{(R,T)}(s, w)$; soit w le plus proche sommet de s ayant cette propriété. Soit P un plus court chemin (resp. une plus courte chaîne) de s à w dans G, et soit $e = (v, w)$ le dernier arc (resp. la dernière arête) de P. $\mathrm{dist}_G(s, v) = \mathrm{dist}_{(R,T)}(s, v)$, mais e n'appartient pas à T. De plus, $l(w) = \mathrm{dist}_{(R,T)}(s, w) > \mathrm{dist}_G(s, w) = \mathrm{dist}_G(s, v) + 1 = \mathrm{dist}_{(R,T)}(s, v) + 1 = l(v) + 1$. Cette inégalité, compte tenu de l'observation précédente, montre que w n'appartenait pas à R quand v a été enlevé de Q, ce qui contredit ③ puisque $e = (v, w)$ existe. □

Ce résultat se déduira également de l'ALGORITHME DE DIJKSTRA pour le PROBLÈME DU PLUS COURT CHEMIN, qu'on peut voir comme une généralisation de BFS dans le cas où les poids des arcs sont non négatifs (voir paragraphe 7.1).

Nous allons maintenant montrer comment trouver les composantes fortement connexes d'un graphe orienté. Bien sûr, cela peut se faire facilement en appliquant n fois DFS ou BFS. Cependant, on peut trouver les composantes fortement connexes en visitant chaque arc deux fois :

ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES	
Input	Un graphe orienté G.
Output	Une fonction $comp : V(G) \to \mathbb{N}$ indiquant le numéro de la composante fortement connexe de chaque sommet.

① $R := \emptyset$. $N := 0$.

② **For** tout $v \in V(G)$ **do** : **If** $v \notin R$ **then** VISIT1(v).

③ $R := \emptyset$. $K := 0$.

④ **For** $i := |V(G)|$ **down to** 1 **do** :
If $\psi^{-1}(i) \notin R$ **then** $K := K + 1$ et VISIT2($\psi^{-1}(i)$).

VISIT1(v)

① $R := R \cup \{v\}$.

② **For** tout $w \in V(G) \setminus R$ avec $(v, w) \in E(G)$ **do** VISIT1(w).

③ $N := N + 1$, $\psi(v) := N$ et $\psi^{-1}(N) := v$.

VISIT2(v)

① $R := R \cup \{v\}$.

② **For** tout $w \in V(G) \setminus R$ avec $(w, v) \in E(G)$ **do** VISIT2(w).

③ Set $comp(v) := K$.

La figure 2.3 montre un exemple : le premier DFS examine les sommets dans l'ordre a, g, b, d, e, f et fournit l'arborescence montrée au milieu ; les nombres sont les labels ψ. Le sommet c est le seul qu'on ne peut atteindre à partir de a ; il obtient le plus grand label $\psi(c) = 7$. Le second DFS part de c, mais ne peut atteindre aucun autre sommet par des arcs inversés. Donc, il poursuit avec le sommet a puisque $\psi(a) = 6$. b, g et f peuvent être alors atteints (toujours avec des arcs inversés). Finalement e est atteint à partir de d. Les composantes fortement connexes sont $\{c\}$, $\{a, b, f, g\}$ et $\{d, e\}$.

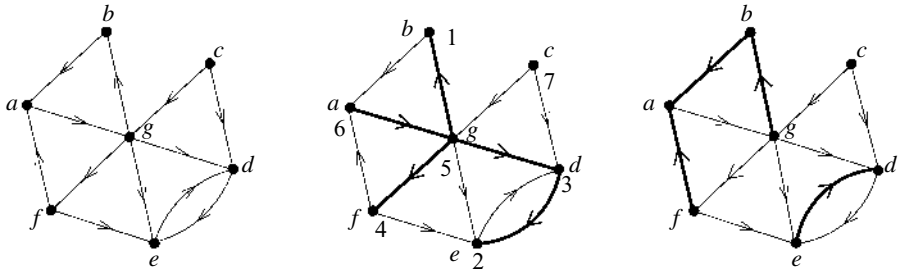


Figure 2.3.

En résumé, un premier DFS est utilisé pour trouver des labels appropriés, tandis que le second DFS considère le graphe où les orientations sont inversées et les sommets sont examinés par ordre décroissant par rapport à l'ordre induit par les labels. Chaque composante connexe de la seconde forêt-DFS est une **antiarborescence**, graphe obtenu en inversant tous les arcs d'une arborescence. Nous montrons maintenant que ces antiarborescences identifient les composantes fortement connexes.

Théorème 2.19. *L'*ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES *identifie les composantes fortement connexes en temps linéaire.*

Preuve. Le temps de calcul est manifestement $O(n+m)$. Les sommets de la même composante fortement connexe sont toujours dans la même composante connexe des deux forêts-DFS et $comp$ leur attribue donc la même valeur. Nous devons montrer que deux sommets u et v pour lesquels $comp(u) = comp(v)$ appartiennent à la même composante fortement connexe. Soient $r(u)$ et $r(v)$ les sommets connectés respectivement à u et v ayant le plus grand label ψ. Puisque $comp(u) = comp(v)$, u et v appartiennent à la même antiarborescence de la seconde forêt-DFS, et $r := r(u) = r(v)$ est la racine de cette antiarborescence. Donc r est connecté à u et à v.

Puisque r est connecté à u et puisque $\psi(r) \geq \psi(u)$, r n'a pas été rajouté à R après u dans le premier DFS, et la première forêt-DFS contient donc un chemin de r à u. En d'autres termes, u est connecté à r. Il en est de même pour v. Ainsi, u

et v sont connectés l'un à l'autre et ils appartiennent bien à la même composante fortement connexe. □

Il est intéressant de noter que cet algorithme résout un autre problème : trouver un ordre topologique dans un graphe sans circuit. Observons que la contraction des composantes fortement connexes d'un graphe orienté permet d'obtenir un graphe orienté sans circuit. Par la proposition 2.9, ce graphe orienté a un ordre topologique. En fait, un tel ordre est obtenu à partir des nombres $comp(v)$ calculés par l'ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES :

Théorème 2.20. *L'*ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES *trouve un ordre topologique du graphe orienté résultant de la contraction de chaque composante fortement connexe de G. Si G est orienté, on peut, en temps linéaire, trouver un ordre topologique ou conclure qu'un tel ordre n'existe pas.*

Preuve. Soient X et Y deux composantes fortement connexes du graphe orienté G, et supposons que l'ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES trouve $comp(x) = k_1$ pour $x \in X$ et $comp(y) = k_2$ pour $y \in Y$ avec $k_1 < k_2$. Montrons qu'on a alors $E_G^+(Y, X) = \emptyset$.

Supposons qu'il existe un arc $(y, x) \in E(G)$ avec $y \in Y$ et $x \in X$. Tous les sommets de X sont ajoutés à R dans le second DFS avant que le premier sommet de Y ne soit ajouté. En particulier $x \in R$ et $y \notin R$ quand l'arc (y, x) est examiné dans le deuxième DFS. Mais cela signifie que y est ajouté à R avant que K soit incrémenté, ce qui contredit $comp(y) \neq comp(x)$.

Ainsi les valeurs de la fonction $comp$ calculées par l'ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES trouve un ordre topologique du graphe orienté résultant de la contraction des composantes fortement connexes. La seconde affirmation du théorème se déduit de la proposition 2.9 et de l'observation suivante : un graphe orienté est sans circuit si et seulement si ses composantes fortement connexes sont des singletons. □

Le premier algorithme fournissant les composantes fortement connexes en temps linéaire a été proposé par Tarjan [1972]. Le problème concernant l'existence d'un ordre topologique a été résolu plus tôt (Kahn [1962], Knuth [1968]). Ces deux algorithmes BFS et DFS sont souvent des sous-programmes dans de nombreux algorithmes combinatoires. Des exemples seront donnés dans les chapitres suivants.

On est intéressé parfois par des notions de connexité plus fortes que les précédentes. Soit $k \geq 2$. Un graphe non orienté ayant au moins k sommets et qui reste connexe quand on retire $k-1$ sommets quelconques sera appelé ***k*-connexe**. Un graphe ayant au moins deux sommets et qui reste connexe quand on supprime $k-1$ arêtes quelconques sera dit ***k*-arête-connexe**

Ainsi un graphe connexe avec au moins trois sommets est 2-connexe (resp. 2-arête-connexe) si et seulement s'il n'a pas de sommet d'articulation (resp. de pont).

Le plus grand k et le plus grand l pour lesquels un graphe G est k-connexe et l-arête-connexe sont appelés le **nombre de connexité** et l'**indice de connexité** de

G. Un graphe est 1-connexe et aussi 1-arête-connexe s'il est connexe. Un graphe non connexe a un nombre de connexité et un index de connexité égaux à zéro.

Les **blocs** d'un graphe non orienté sont les sous-graphes connexes maximaux sans sommets d'articulation. Ainsi un bloc est soit un sous-graphe 2-connexe maximal, soit un pont, soit un sommet isolé. Deux blocs ont au plus un sommet en commun, et un sommet qui appartient à plus qu'un bloc est un sommet d'articulation. Les blocs d'un graphe non orienté peuvent être trouvés en temps linéaire de la même manière que dans l'ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES ; voir exercice 16. Les graphes 2-connexes possèdent une structure intéressante que nous allons étudier. Nous construisons ces graphes en partant d'un sommet et en ajoutant de manière séquentielle des oreilles :

Définition 2.21. *Soit G un graphe (orienté ou non orienté). Une* **décomposition en oreilles** *de G est une suite $r, P_1, \ldots, P_k$ avec $G = (\{r\}, \emptyset) + P_1 + \cdots + P_k$, telle que chaque P_i est soit une chaîne, soit un chemin dont seules les extrémités appartiennent à $\{r\} \cup V(P_1) \cup \cdots \cup V(P_{i-1})$, ou un cycle ou un circuit n'ayant qu'un seul sommet dans $\{r\} \cup V(P_1) \cup \cdots \cup V(P_{i-1})$ ($i \in \{1, \ldots, k\}$).*

$P_1, \ldots, P_k$ sont appelées **oreilles***. Si $k \geq 1$, si P_1 est un cycle ou un circuit avec au moins trois sommets et si $P_2, \ldots, P_k$ sont des chaînes ou chemins, on dit alors que la décomposition en oreilles est* **propre***.*

Théorème 2.22. (Whitney [1932]) *Un graphe non orienté est 2-connexe si et seulement s'il possède une décomposition en oreilles propre.*

Preuve. Un cycle de longueur au moins trois est 2-connexe. De plus, si G est 2-connexe, $G + P$, où P est une chaîne reliant deux sommets distincts x, y de G est aussi 2-connexe puisqu'on ne détruit pas la connexité par suppression d'un sommet. Un graphe ayant une décomposition en oreilles propre est donc 2-connexe.

Pour montrer la réciproque, soit G un graphe 2-connexe. Soit G' un sous-graphe simple maximal de G ; G' est aussi 2-connexe. Donc G' n'est pas un arbre et contient un cycle et même un cycle de longueur au moins trois puisqu'il est simple. Soit H un sous-graphe maximal de G ayant une décomposition propre en oreilles.

Supposons que H ne soit pas couvrant. Puisque G est connexe, nous savons qu'il existe une arête $e = (x, y) \in E(G)$ avec $x \in V(H)$ et $y \notin V(H)$. Soit z un sommet de $V(H) \setminus \{x\}$. Puisque $G - x$ est connexe, il existe une chaîne P de y à z dans $G - x$. Décrivant cette chaîne à partir de y, soit z' le premier sommet rencontré qui appartient à $V(H)$. Alors $P_{[y,z']} + e$ peut être rajouté, en tant qu'oreille contredisant la maximalité de H.

Donc H est couvrant. Puisque chaque arête de $E(G) \setminus E(H)$ peut être rajoutée en tant qu'oreille, nous concluons que $H = G$. □

Voir l'exercice 17 pour des caractérisations similaires des graphes 2-arête-connexes et des graphes orientés fortement connexes.

2.4 Graphes eulériens et bipartis

On attribue souvent l'origine de la théorie des graphes à Euler qui cherchait à savoir s'il existait un parcours empruntant une fois et une seule chacun des sept ponts de la ville de Königsberg. Euler a montré qu'un tel parcours était impossible en construisant un graphe ayant plus de deux sommets de degré impair.

Définition 2.23. *Un* **parcours eulérien** *(resp. un* **parcours orienté eulérien***) dans un graphe G non orienté (resp. orienté) est un parcours fermé (resp. un parcours orienté fermé) qui contient chaque arête (resp. arc) de G. Un graphe non orienté G est* **eulérien** *si le degré de chaque sommet est pair. Un graphe orienté G est* **eulérien** *si $|\delta^-(v)| = |\delta^+(v)|$ pour tout $v \in V(G)$.*

Bien qu'Euler n'ait jamais établi explicitement le théorème suivant, la tradition lui en attribue la paternité :

Théorème 2.24. (Euler [1736], Hierholzer [1873]) *Un graphe connexe a un parcours eulérien (resp. un parcours orienté eulérien) si et seulement s'il est eulérien.*

Preuve. La nécessité est évidente ; la condition suffisante est montrée par l'algorithme suivant (théorème 2.25). □

L'input de l'algorithme est un graphe connexe eulérien. Remarquons qu'on peut vérifier en temps linéaire si un graphe est connexe (théorème 2.17) et eulérien. L'algorithme choisit d'abord un sommet initial puis appelle une procédure récursive. Plaçons-nous d'abord dans le cas des graphes non orientés :

ALGORITHME D'EULER	
Input	Un graphe non orienté connexe eulérien G.
Output	Un parcours eulérien W dans G.

① Choisir $v_1 \in V(G)$ arbitrairement. **Retourner** $W :=$ EULER(G, v_1).

EULER(G, v_1)

① $W := v_1$ et $x := v_1$.

② **If** $\delta(x) = \emptyset$ **then go to** ④.
Else soit $e \in \delta(x)$; on peut supposer que $e = (x, y)$.

③ $W := W, e, y$ et $x := y$. $E(G) := E(G) \setminus \{e\}$ et **go to** ②.

④ Soit $v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}$ la suite W.
For $i := 1$ **to** k **do** : $W_i :=$ EULER(G, v_i).

⑤ Set $W := W_1, e_1, W_2, e_2, \ldots, W_k, e_k, v_{k+1}$. **Retourner** W.

Pour les graphes orientés, remplacer ② par :

② **If** $\delta^+(x) = \emptyset$ **then go to** ④.
Else soit $e \in \delta^+(x)$; on peut supposer que $e = (x, y)$.

Nous allons analyser les deux versions (orientée ou non) simultanément :

Théorème 2.25. L'ALGORITHME D'EULER *répond correctement. Il s'exécute en un temps* $O(m + n)$ *(*$n = |V(G)|$*,* $m = |E(G)|$*).*

Preuve. Par induction sur $|E(G)|$, le cas $E(G) = \emptyset$ étant trivial.

Les conditions de degré impliquent que $v_{k+1} = x = v_1$ quand on entre dans ④. W est alors un parcours orienté fermé (resp. un parcours fermé) et G' obtenu à cette étape vérifie aussi les contraintes de degré.

Pour chaque arc (resp. arête) $e \in E(G')$ soit $i \in \{1, \ldots, k\}$ l'indice minimum tel que e et v_i soient dans la même composante connexe de G'. Par l'hypothèse d'induction e appartient à W_i. Donc la suite W de ⑤ est bien un parcours orienté (resp. un parcours) eulérien.

Le temps de calcul est linéaire, puisque chaque arc ou arête est immédiatement effacé après avoir été examiné. □

L'ALGORITHME D'EULER sera souvent utilisé comme sous-programme au cours des chapitres suivants.

On pourra parfois chercher à rendre un graphe eulérien en ajoutant ou en contractant des arêtes. Soient G un graphe non orienté et F une famille de paires de sommets : F est un **joint impair** si $(V(G), E(G) \dot\cup F)$ est eulérien et F est une **couverture impaire** si le graphe obtenu en contractant successivement chaque arête $e \in F$ de G est eulérien. Les deux concepts sont en réalité équivalents :

Théorème 2.26. (Aoshima et Iri [1977]) *Soit* G *un graphe non orienté :*

(a) *Tout joint impair est une couverture impaire.*

(b) *Toute couverture impaire minimale est un joint impair.*

Preuve. Soit G un graphe non orienté.

Pour montrer (a), soit F un joint impair et soit G' le graphe obtenu en contractant les composantes connexes de $(V(G), F)$ dans G. Chacune de ces composantes connexes contient un nombre pair de sommets de degré impair dans le graphe $(V(G), F)$ et donc aussi dans le graphe G, puisque F est un joint impair. Donc tous les degrés du graphe résultant sont pairs et F est une couverture impaire.

Pour montrer (b), soit F une couverture impaire minimale. La condition de minimalité implique que $(V(G), F)$ est une forêt. Il suffit de montrer que $|\delta_F(v)| \equiv |\delta_G(v)| \pmod 2$ pour tout $v \in V(G)$. Soit $v \in V(G)$ et soient $C_1, \ldots, C_k$ les composantes connexes de $(V(G), F) - v$ qui contiennent un sommet w avec $(v, w) \in F$. Puisque F est une forêt, $k = |\delta_F(v)|$.

Comme F est une couverture impaire, la contraction de $X := V(C_1) \cup \cdots \cup V(C_k) \cup \{v\}$ dans G crée un sommet de degré pair et $|\delta_G(X)|$ est pair. D'autre part la minimalité de F implique que $F \setminus \{(v, w)\}$ n'est pas une couverture impaire (quel que soit w avec $(v, w) \in F$) et donc $|\delta_G(V(C_i))|$ est impair pour $i = 1, \ldots, k$. Puisque

$$\sum_{i=1}^{k} |\delta_G(V(C_i))| = |\delta_G(X)| + |\delta_G(v)| - 2|E_G(\{v\}, V(G) \setminus X)| + 2 \sum_{1 \leq i < j \leq k} |E_G(C_i, C_j)|,$$

k a la même parité que $|\delta_G(v)|$. □

Nous verrons comment rendre un graphe eulérien au paragraphe 12.2.

Une **bipartition** d'un graphe non orienté G est une partition de l'ensemble des sommets $V(G) = A \dot\cup B$ pour laquelle les sous-graphes induits par A et B sont vides. Nous dirons qu'un graphe est **biparti** s'il possède une bipartition. Le graphe simple G tel que $V(G) = A \dot\cup B$, $|A| = n$, $|B| = m$ et $E(G) = \{(a, b) : a \in A, b \in B\}$ sera noté $K_{n,m}$ (le graphe complet biparti). Quand nous écrirons $G = (A \dot\cup B, E(G))$, cela signifiera que $G[A]$ et $G[B]$ sont vides.

Proposition 2.27. (König [1916]) *Un graphe non orienté est biparti si et seulement s'il ne contient aucun cycle impair. Il existe un algorithme linéaire qui, étant donné un graphe non orienté, trouve soit une bipartition, soit un cycle impair.*

Preuve. Soit G biparti avec une bipartition $V(G) = A \dot\cup B$.
Si $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ est un cycle de G, on peut toujours supposer que $v_1 \in A$. Mais alors $v_2 \in B$, $v_3 \in A$, et ainsi de suite. Donc $v_i \in A$ si et seulement si i est impair. Mais comme $v_{k+1} = v_1 \in A$, k est pair.

Pour montrer la condition suffisante, on peut supposer G connexe, puisqu'un graphe est biparti si et seulement si ses composantes connexes sont biparties (de plus, les composantes connexes peuvent être trouvées en temps linéaire ; voir la proposition 2.17). Choisissons un sommet arbitraire $s \in V(G)$ et appliquons BFS à (G, s) de manière à obtenir les distances de s à tout v $v \in V(G)$ (voir proposition 2.18). Soit T l'arbre-BFS ainsi obtenu. Posons $A := \{v \in V(G) : \mathrm{dist}_G(s, v)$ est pair$\}$ et $B := V(G) \setminus A$.

S'il existe une arête $e = (x, y)$ dans $G[A]$ ou $G[B]$, la chaîne de x à y dans T à laquelle on ajoute e est un cycle impair de G. S'il n'existe aucune arête de ce type, nous obtenons une bipartition. □

2.5 Planarité

Nous dessinons souvent des graphes sur un plan. Un graphe est planaire si on peut le dessiner dans le plan de telle sorte que ses arêtes ne se croisent pas. Pour formaliser ce concept rappelons quelques définitions topologiques :

Définition 2.28. *Une* **courbe de Jordan simple** *est l'image d'une fonction continue injective $\varphi : [0, 1] \to \mathbb{R}^2$; ses* **extrémités** *sont $\varphi(0)$ et $\varphi(1)$. Une* **courbe de Jordan fermée** *est l'image d'une fonction continue $\varphi : [0, 1] \to \mathbb{R}^2$ avec $\varphi(0) = \varphi(1)$ et $\varphi(\tau) \neq \varphi(\tau')$ pour $0 \leq \tau < \tau' < 1$. Une* **courbe polygonale**

est une courbe de Jordan simple qui est l'union d'un nombre fini d'intervalles (segments de droite). Un **polygone** *est une courbe de Jordan fermée qui est l'union d'un nombre fini d'intervalles.*

Soit $R = \mathbb{R}^2 \setminus J$, où J est l'union d'un nombre fini d'intervalles. Une **région connexe** *de R est une classe d'équivalence, deux points de R étant équivalents s'ils peuvent être joints par une courbe polygonale contenue dans R.*

Définition 2.29. *Une* **représentation plane** *d'un graphe G consiste en une application injective $\psi : V(G) \to \mathbb{R}^2$ et en des courbes polygonales J_e d'extrémités $\psi(x)$ et $\psi(y)$ associées à chaque arête $e = (x, y) \in E(G)$ de telle sorte que :*

$$(J_e \setminus \{\psi(x), \psi(y)\}) \cap \left(\{\psi(v) : v \in V(G)\} \cup \bigcup_{e' \in E(G) \setminus \{e\}} J_{e'} \right) = \emptyset.$$

Un graphe est dit **planaire** *si on peut lui associer une représentation plane.*

Soit G un graphe planaire et une représentation plane $\Phi = (\psi, (J_e)_{e \in E(G)})$ de ce graphe. Quand on retire du plan les points et les courbes polygonales, le reste du plan

$$R := \mathbb{R}^2 \setminus \left(\{\psi(v) : v \in V(G)\} \cup \bigcup_{e \in E(G)} J_e \right)$$

se sépare en régions connexes, appelées **faces** *de Φ.*

Par exemple, K_4 est planaire, mais on montrera que K_5 ne l'est pas. L'exercice 23 montre qu'on peut toujours se restreindre aux courbes polygonales plutôt qu'aux courbes de Jordan générales. Nous montrerons que, pour les graphes simples, on peut considérer que les courbes polygonales sont des segments de droite.

Notre objectif est de caractériser les graphes planaires. Suivant Thomassen [1981], prouvons d'abord le résultat topologique suivant qui est une version du théorème sur les courbes de Jordan :

Théorème 2.30. *Soit J un polygone ; alors $\mathbb{R}^2 \setminus J$ se décompose en deux régions connexes, chacune d'entre elles ayant J comme bord. Si J est une courbe polygonale, alors $\mathbb{R}^2 \setminus J$ n'a qu'une région connexe.*

Preuve. Soit J un polygone, $p \in \mathbb{R}^2 \setminus J$ et $q \in J$. Alors il existe une courbe polygonale $(\mathbb{R}^2 \setminus J) \cup \{q\}$ joignant p et q : partant de p, on se déplace vers q en empruntant le segment de droite d'extrémités p et q ; quand on atteint le voisinage de J, on se déplace vers q, en ne sortant pas de ce voisinage. (Nous utilisons ici le résultat topologique élémentaire suivant : des ensembles compacts disjoints, en particulier des intervalles non adjacents de J, ont entre eux des distances positives.) Nous en déduisons que p est dans la même région connexe de $\mathbb{R}^2 \setminus J$ que certains points arbitrairement proches de q.

J est l'union d'un nombre fini d'intervalles ; un ou deux de ces intervalles contiennent q. Soit $\epsilon > 0$ tel que la boule de centre q et de rayon ϵ ne contienne

aucun autre intervalle de J ; cette boule intersecte au plus deux régions connexes de $\mathbb{R}^2 \setminus J$. Puisque $p \in \mathbb{R}^2 \setminus J$ et $q \in J$ ont été choisis arbitrairement, nous en déduisons qu'il existe au plus deux régions connexes et que J est la frontière (ou bord) de chacune de ces deux régions.

Si J est une courbe polygonale et q est une de ses extrémités, on voit alors que dans ce cas, $\mathbb{R}^2 \setminus J$ n'a qu'une seule région.

Dans le cas où J est un polygone, il nous reste à montrer que $\mathbb{R}^2 \setminus J$ a exactement deux régions. Pour tout $p \in \mathbb{R}^2 \setminus J$ et tout angle α considérons le rayon l_α d'origine p et d'angle α. $J \cap l_\alpha$ est un ensemble de points ou d'intervalles fermés. Soit $cr(p, l_\alpha)$ le nombre de ces points ou intervalles pour lesquels J pénètre et quitte l_α par des côtés différents de l_α (le nombre de passages de J «à travers» l_α ; par exemple, sur la figure 2.4 $cr(p, l_\alpha) = 2$).

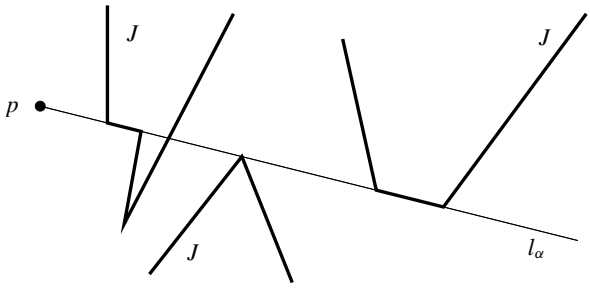


Figure 2.4.

Remarquons que pour chaque α,

$$\left| \lim_{\epsilon \to 0,\, \epsilon > 0} cr(p, l_{\alpha+\epsilon}) - cr(p, l_\alpha) \right| + \left| \lim_{\epsilon \to 0,\, \epsilon < 0} cr(p, l_{\alpha+\epsilon}) - cr(p, l_\alpha) \right|$$

est égal à deux fois le nombre d'intervalles de $J \cap l_\alpha$ pour lesquels J pénètre et quitte l_α par le même côté. Donc $g(p, \alpha) := (cr(p, l_\alpha) \bmod 2)$ est une fonction continue par rapport à α ; c'est donc une constante que nous noterons $g(p)$. Il est évident que $g(p)$ reste constant pour tous les points p d'une même ligne droite n'intersectant pas J, et est donc constant dans chaque région. Cependant, $g(p) \neq g(q)$ pour des points p, q tels que le segment d'extrémités p et q intersecte J une fois exactement. Il y a donc bien deux régions. □

Une seule des faces, la **face extérieure**, est non bornée.

Proposition 2.31. *Soit G un graphe 2-connexe ayant une représentation plane Φ. Chaque face est bordée par un cycle, et chaque arête appartient au bord de deux faces. De plus, le nombre de faces est $|E(G)| - |V(G)| + 2$.*

Preuve. Par le théorème 2.30, ces deux propositions sont vraies si G est un cycle. Pour les graphes 2-connexes quelconques nous ferons une induction sur le nombre

d'arêtes, en utilisant le théorème 2.22. Considérons une décomposition propre en oreilles de G, et soient x et y les extrémités de la dernière oreille P (qui est une chaîne). Soit G' le graphe avant l'ajout de P, et soit Φ' la restriction de Φ à G'.

Soit $\Phi = (\psi, (J_e)_{e\in E(G)})$. Soit F' la face de Φ' contenant $\bigcup_{e\in E(P)} J_e \setminus \{\psi(x), \psi(y)\}$. Par induction, F' est bordée par un cycle C. C contient x et y, et C est donc l'union de deux chaînes Q_1, Q_2 de x à y dans G'. Appliquons alors le théorème 2.30 à chacun des cycles $Q_1 + P$ et $Q_2 + P$. On a alors

$$F' \cup \{\psi(x), \psi(y)\} \;=\; F_1 \,\dot\cup\, F_2 \,\dot\cup \bigcup_{e\in E(P)} J_e$$

et F_1 et F_2 sont deux faces de G bordées, respectivement, par les cycles $Q_1 + P$ et $Q_2 + P$. Donc G a une face de plus que G'. Comme $|E(G) \setminus E(G')| = |V(G) \setminus V(G')| + 1$, cela montre la proposition par induction. □

Cette preuve (de Tutte) implique également que les cycles bordant les faces bornées forment une base des cycles (voir exercice 24). La dernière condition de la proposition 2.31 est connue sous le nom de formule d'Euler ; elle est vraie pour les graphes connexes :

Théorème 2.32. (Euler [1758], Legendre [1794]) *Si G est un graphe planaire connexe, le nombre de faces dans une représentation plane est : $|E(G)|-|V(G)|+2$.*

Preuve. Nous avons montré ce résultat pour les graphes 2-connexes (proposition 2.31). De plus, le résultat est trivial si $|V(G)| = 1$ et se déduit du théorème 2.30 si $|E(G)| = 1$. Si $|V(G)| = 2$ et $|E(G)| \geq 2$, nous pouvons subdiviser une arête e, ajoutant ainsi une arête, un sommet et en rendant le graphe 2-connexe ; la conclusion se déduit de la proposition 2.31.

Nous pouvons donc supposer que G a un sommet d'articulation x ; procédons alors par induction sur le nombre de sommets. Soit Φ une représentation plane de G. Soient $C_1, \ldots, C_k$ les composantes connexes de $G - x$; et soit Φ_i la restriction de Φ à $G_i := G[V(C_i) \cup \{x\}]$ pour $i = 1, \ldots, k$.

L'ensemble des faces intérieures (c.-à-d. bornées) de Φ est l'union disjointe des ensembles des faces intérieures de Φ_i, $i = 1, \ldots, k$. Notre hypothèse d'induction étant vraie pour (G_i, Φ_i), $i = 1, \ldots, k$, le nombre total de faces intérieures de (G, Φ) est

$$\sum_{i=1}^{k}(|E(G_i)|-|V(G_i)|+1) \;=\; |E(G)|-\sum_{i=1}^{k}|V(G_i)\setminus\{x\}| \;=\; |E(G)|-|V(G)|+1.$$

Il suffit de rajouter la face extérieure pour conclure. □

En particulier, le nombre de faces est indépendant de la représentation plane choisie. Le degré moyen d'un graphe simple planaire est inférieur à 6 :

Corollaire 2.33. *Soit G un graphe simple planaire 2-connexe et supposons que la longueur du cycle minimum de G soit k (k est aussi appelé la* **maille** *de G). Alors*

G a au plus $(n-2)\frac{k}{k-2}$ arêtes. Tout graphe planaire ayant $n \geq 3$ sommets a au plus $3n-6$ arêtes.

Preuve. Supposons d'abord que G soit 2-connexe. Soit Φ une représentation plane de G et soit r le nombre de faces. Par la formule d'Euler (théorème 2.32), $r = |E(G)| - |V(G)| + 2$. Par la proposition 2.31, chaque face est bordée par un cycle, c.-à-d. par au moins k arêtes, et chaque arête est sur le bord de deux faces. Donc $kr \leq 2|E(G)|$. Combinant ces deux résultats nous obtenons $|E(G)| - |V(G)| + 2 \leq \frac{2}{k}|E(G)|$, ce qui implique $|E(G)| \leq (n-2)\frac{k}{k-2}$.

Si G n'est pas 2-connexe, nous ajoutons des arêtes entre des sommets non adjacents jusqu'à ce qu'on obtienne un graphe 2-connexe planaire. Par la première partie, il y a au plus $(n-2)\frac{3}{3-2}$ arêtes, en incluant les arêtes ajoutées. □

Nous allons maintenant montrer que certains graphes sont non planaires :

Corollaire 2.34. *Ni K_5 ni $K_{3,3}$ ne sont planaires.*

Preuve. C'est une conséquence directe du corollaire 2.33 : K_5 a cinq sommets, mais $10 > 3 \cdot 5 - 6$ arêtes ; $K_{3,3}$ est 2-connexe, sa maille est 4 (puisqu'il est biparti) et $9 > (6-2)\frac{4}{4-2}$ arêtes. □

Figure 2.5.

La figure 2.5 montre ces deux graphes, qui sont les plus petits graphes non planaires. Nous allons montrer que tout graphe non planaire contient, d'une certaine manière, K_5 ou $K_{3,3}$. Afin de préciser cela, introduisons la notion suivante :

Définition 2.35. *Soient G et H deux graphes non orientés. G est un* **mineur** *de H s'il existe un sous-graphe H' de H et une partition $V(H') = V_1 \dot\cup \cdots \dot\cup V_k$ de l'ensemble de ses sommets en sous-ensembles connexes de telle sorte que le graphe obtenu par contraction de $V_1, \ldots, V_k$ soit isomorphe à G.*

Autrement dit, G est un mineur de H si on peut obtenir G à partir de H par une succession d'opérations du type suivant : enlever un sommet, enlever une arête ou contracter une arête. Puisque aucune de ces opérations ne détruit la planarité, tout mineur d'un graphe planaire est planaire. Donc un graphe qui contient K_5 ou $K_{3,3}$ comme mineur ne peut être planaire. Le théorème de Kuratowski affirme que la

réciproque est également vraie. Considérons d'abord le cas des graphes 3-connexes et établissons le lemme suivant (qui est le fondement du théorème de la roue connu sous le nom de «*wheel theorem*» de Tutte) :

Lemme 2.36. (Tutte [1961], Thomassen [1980]) *Soit G un graphe 3-connexe ayant au moins cinq sommets. Il existe une arête e telle que G/e soit 3-connexe.*

Preuve. Supposons ce résultat faux. Alors, pour chaque arête $e = (v, w)$ il existe un sommet x tel que $G - \{v, w, x\}$ soit non connexe, c.-à-d. a une composante connexe C avec $|V(C)| < |V(G)| - 3$. Choisissons e, x et C de telle sorte que $|V(C)|$ soit minimum.

x a un voisin y dans C, sinon C serait une composante connexe de $G - \{v, w\}$ (mais G est 3-connexe). Par notre hypothèse, $G/\{x, y\}$ n'est pas 3-connexe, c.-à-d. qu'il existe un sommet z tel que $G - \{x, y, z\}$ est non connexe. Puisque $(v, w) \in E(G)$, il existe une composante connexe D de $G - \{x, y, z\}$ qui ne contient ni v ni w.

Mais D contient un voisin d de y, car autrement D serait une composante connexe de $G - \{x, z\}$ (contredisant encore le fait que G est 3-connexe). Ainsi $d \in V(D) \cap V(C)$, et D est un sous-graphe de C. Puisque $y \in V(C) \setminus V(D)$, cela contredit la minimalité de $|V(C)|$. □

Théorème 2.37. (Kuratowski [1930], Wagner [1937]) *Un graphe 3-connexe est planaire si et seulement s'il ne contient ni K_5 ni $K_{3,3}$ comme mineur.*

Preuve. Par ce qui précède, il nous reste à montrer que la condition est suffisante. Puisque K_4 est planaire, nous procéderons par induction sur le nombre de sommets : soit G un graphe 3-connexe avec plus de quatre sommets, mais sans K_5 ni $K_{3,3}$ comme mineur.

Par le lemme 2.36, il existe une arête $e = (v, w)$ telle que G/e soit 3-connexe. Soit $\Phi = \big(\psi, (J_e)_{e \in E(G)}\big)$ une représentation plane de G/e, qui existe par induction. Soit x le sommet dans G/e résultant de la contraction de e. Retirons x de (G/e) et de sa représentation plane. Puisque $(G/e) - x$ est 2-connexe, toute face est bordée par un cycle (proposition 2.31). En particulier, la face contenant le point $\psi(x)$ est bordée par un cycle C.

Soient $y_1, \ldots, y_k \in V(C)$ les voisins de v distincts de w, énumérés dans l'ordre induit par un des deux parcours du cycle et partitionnons C en chaînes arête-disjointes P_i, $i = 1, \ldots, k$, de telle sorte que P_i soit une chaîne de y_i à y_{i+1} ($y_{k+1} := y_1$).

Supposons qu'il existe un indice $i \in \{1, \ldots, k\}$ tel que $\Gamma(w) \subseteq \{v\} \cup V(P_i)$. Il est alors facile de réaliser une représentation plane de G en modifiant Φ.

Montrons que les autres cas sont impossibles. Si w a trois voisins parmi les sommets $y_1, \ldots, y_k$, nous obtenons un mineur K_5 (figure 2.6(a)).

Si $\Gamma(w) = \{v, y_i, y_j\}$ avec $i < j$, alors $i + 1 < j$ et $(i, j) \neq (1, k)$ (sinon y_i et y_j appartiendraient tous deux à P_i ou à P_j ; voir la figure 2.6(b)). Sinon il existe un voisin z de w dans $V(P_i) \setminus \{y_i, y_{i+1}\}$ et un voisin $z' \notin V(P_i)$ (figure 2.6(c)).

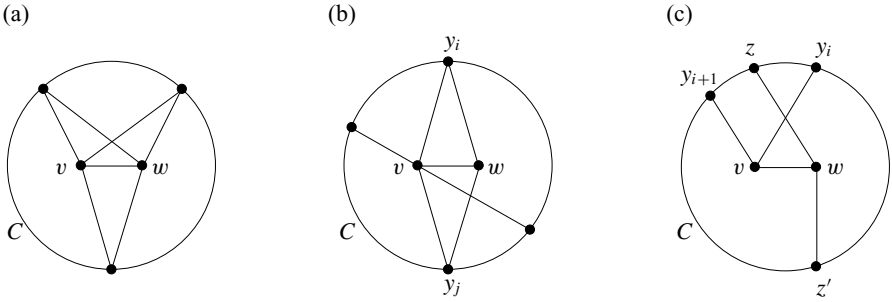


Figure 2.6.

Dans les deux cas, il existe quatre sommets y, z, y', z' dans cet ordre sur C, avec $y, y' \in \Gamma(v)$ et $z, z' \in \Gamma(w)$. Cela implique qu'il existe un mineur $K_{3,3}$. □

Cette démonstration montre également que tout graphe planaire simple 3-connexe a une représentation plane où chaque arête est représentée par un segment de droite et où chaque face à l'exception de la face externe est convexe (voir exercice 27(a)). Le cas général du théorème de Kuratowski peut se ramener au cas 3-connexe grâce au lemme suivant :

Lemme 2.38. (Thomassen [1980]) *Soit G un graphe ayant au moins cinq sommets qui n'est pas 3-connexe et qui ne contient ni K_5 ni $K_{3,3}$ comme mineur. Il existe deux sommets non adjacents $v, w \in V(G)$ et une nouvelle arête $e = (v, w)$ de telle sorte que $G + e$ ne contienne ni K_5 ni $K_{3,3}$ comme mineur.*

Preuve. Par induction sur $|V(G)|$. Si G n'est pas connexe, nous pouvons ajouter une arête e joignant deux composantes connexes et le lemme est démontré. Supposons donc G connexe. Puisque G n'est pas 3-connexe, il existe un ensemble de deux sommets $X = \{x, y\}$ tel que $G - X$ ne soit pas connexe. (Si G n'est pas 2-connexe nous pouvons choisir pour x un sommet d'articulation et pour y un voisin de x.) Soient C une composante connexe de $G - X$, $G_1 := G[V(C) \cup X]$ et $G_2 := G - V(C)$. Montrons d'abord le résultat intermédiaire suivant : soient deux sommets $v, w \in V(G_1)$ tels que l'addition d'une arête $e = (v, w)$ à G crée un mineur $K_{3,3}$ ou K_5. Alors au moins un des deux graphes $G_1 + e + f$, $G_2 + f$ contient K_5 ou $K_{3,3}$ comme mineur, f étant une nouvelle arête joignant x et y.

Pour montrer cela, soient $v, w \in V(G_1)$, $e = (v, w)$ et supposons qu'il existe des ensembles disjoints de sommets connexes $Z_1, \ldots, Z_t$ de $G+e$ tels que le graphe obtenu après contraction de chacun de ces ensembles contienne K_5 ($t = 5$) ou $K_{3,3}$ ($t = 6$) comme sous-graphe.

On ne peut avoir simultanément $Z_i \subseteq V(G_1) \setminus X$ et $Z_j \subseteq V(G_2) \setminus X$ pour un couple d'indices $i, j \in \{1, \ldots, t\}$, car les ensembles Z_k pour lesquels $Z_k \cap X \neq \emptyset$ (il y en a au plus deux) sépareraient alors Z_i et Z_j, ce qui est impossible K_5 et $K_{3,3}$ étant 3-connexes.

Il y a donc deux cas possibles : si aucun des $Z_1, \ldots, Z_t$ n'est un sous ensemble de $V(G_2) \setminus X$, alors il suffit de contracter $Z_i \cap V(G_1)$ ($i = 1, \ldots, t$) pour obtenir

un des deux mineurs K_5 ou $K_{3,3}$ dans $G_1 + e + f$: de la même manière, si aucun des $Z_1, \ldots, Z_t$ n'est un sous-ensemble de $V(G_1) \setminus X$, alors $G_2 + f$ contient K_5 ou $K_{3,3}$ comme mineur (il suffit de considérer $Z_i \cap V(G_2)$ $(i = 1, \ldots, t)$).

Ce résultat étant établi, étudions d'abord le cas où G contient un sommet d'articulation x ; soient y et z deux voisins de x choisis dans des composantes connexes distinctes de $G - x$. On peut toujours supposer que $z \in V(G_1)$. Si l'addition de $e = (y, z)$ crée un mineur K_5 ou $K_{3,3}$, le résultat précédent implique que, au moins un des deux graphes $G_1 + e$, G_2 contient K_5 ou $K_{3,3}$ comme mineur (l'arête f n'est pas à introduire puisque l'arête (x, y) existe déjà). Mais alors G_1 ou G_2, et donc G, contient K_5 ou $K_{3,3}$ comme mineur, ce qui contredit notre hypothèse.

On peut donc supposer que G est 2-connexe. Rappelons que $G - \{x, y\}$ n'est pas connexe. Si $(x, y) \notin E(G)$ ajoutons simplement une nouvelle arête $f = (x, y)$. Si cette adjonction crée un mineur K_5 ou $K_{3,3}$, le résultat intermédiaire implique que $G_1 + f$ ou $G_2 + f$ contient un tel mineur. Puisqu'il existe une chaîne de x à y dans chacun des graphes G_1, G_2 (sinon G aurait un sommet d'articulation), cela implique que G contient un mineur K_5 ou $K_{3,3}$, ce qui est une contradiction.

Nous pouvons donc supposer que $f = (x, y) \in E(G)$. Si un des deux graphes G_i $(i \in \{1, 2\})$ n'est pas planaire, ce graphe G_i a au moins cinq sommets. Puisqu'il ne contient ni K_5 ni $K_{3,3}$ comme mineur (car ce seraient aussi des mineurs de G), nous pouvons conclure par le théorème de 2.37 que G_i n'est pas 3-connexe. Appliquons alors l'hypothèse d'induction à G_i. Si l'addition d'une arête à G_i ne crée pas de mineur K_5 ou $K_{3,3}$ dans G_i, elle ne crée pas non plus de mineur dans G, par le résultat intermédiaire.

Nous pouvons donc supposer que G_1 et G_2 sont planaires ; soient Φ_1 et Φ_2 deux représentations planes de ces deux graphes. Soit F_i une face de Φ_i ayant f sur son bord et soit z_i un autre sommet sur le bord de F_i, $z_i \notin \{x, y\}$ $(i = 1, 2)$. Alors montrons que l'addition d'une arête (z_1, z_2) (voir figure 2.7) ne crée pas de mineur K_5 ou $K_{3,3}$.

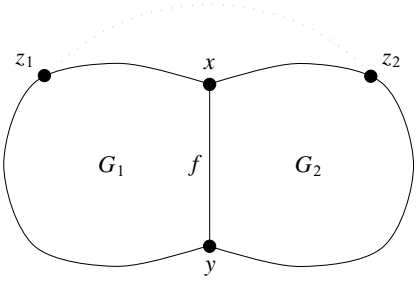


Figure 2.7.

Supposons, au contraire, que l'addition de (z_1, z_2) suivie de la contraction d'ensembles de sommets connexes disjoints $Z_1, \ldots, Z_t$ crée un sous-graphe K_5 $(t = 5)$ ou $K_{3,3}$ $(t = 6)$.

Si au plus un des ensembles Z_i est un sous-ensemble de $V(G_1) \setminus \{x, y\}$, alors le graphe G_2', obtenu à partir de G_2 en ajoutant un sommet w et des arêtes joignant w à x, y et z_2, contient également un mineur K_5 ou $K_{3,3}$. (Ici w correspond à l'ensemble contracté $Z_i \subseteq V(G_1) \setminus \{x, y\}$.) Cela est une contradiction puisqu'il existe une représentation plane de G_2' : il suffit en effet d'étendre Φ_2 en plaçant w à l'intérieur de F_2.

Nous pouvons donc supposer que $Z_1, Z_2 \subseteq V(G_1) \setminus \{x, y\}$.Nous pouvons de même supposer que $Z_3, Z_4 \subseteq V(G_2) \setminus \{x, y\}$. Sans perte de généralités, nous avons $z_1 \notin Z_1$ et $z_2 \notin Z_3$. Nous ne pouvons créer K_5, car Z_1 et Z_3 ne sont pas adjacents. De plus, les seuls voisins communs de Z_1 et Z_3 sont Z_5 et Z_6. Puisque dans $K_{3,3}$ deux sommets sont adjacents ou ont trois voisins communs, il est impossible d'obtenir un mineur $K_{3,3}$. □

Le théorème 2.37 et le lemme 2.38 conduisent au théorème de Kuratowski :

Théorème 2.39. (Kuratowski [1930], Wagner [1937]) *Un graphe non orienté est planaire si et seulement s'il ne contient ni K_5 ni $K_{3,3}$ comme mineur.* □

Kuratowski a, en réalité, démontré un résultat plus fort (voir exercice 28). On peut, en adaptant cette preuve, facilement montrer l'existence d'un algorithme polynomial de planarité (voir exercice 27(b)). Il existe même un algorithme linéaire pour dessiner un graphe dans le plan :

Théorème 2.40. (Hopcroft et Tarjan [1974]) *Il existe un algorithme linéaire pour trouver une représentation plane d'un graphe ou pour décider que ce graphe n'est pas planaire.*

2.6 Dualité planaire

Nous allons introduire maintenant un important concept de dualité. Ce paragraphe est le seul du livre où nous aurons besoin des boucles, c.-à-d. des arêtes dont les deux extrémités sont identiques. Dans une représentation plane, les boucles sont représentées par des polygones à la place de courbes polygonales.

Notons que la formule d'Euler (théorème 2.32) est également vérifiée pour les graphes avec boucles : cela se voit en remarquant qu'en subdivisant une boucle e (c.-à-d. en remplaçant $e = (v, v)$ par deux arêtes parallèles $(v, w), (w, v)$, w étant un nouveau sommet) et en ajustant la représentation plane (c.-à-d. en remplaçant le polygone J_e par deux courbes polygonales dont l'union est J_e) on augmente le nombre d'arêtes et le nombre de sommets d'une unité sans changer le nombre de faces.

Définition 2.41. *Soit G un graphe orienté ou non orienté, avec, éventuellement, des boucles et soit $\Phi = (\psi, (J_e)_{e \in E(G)})$ une représentation plane de G. Le* **graphe planaire dual** *G^* est le graphe dont les sommets sont les faces de Φ et dont l'ensemble des arêtes est $\{e^* : e \in E(G)\}$, où e^* relie les faces incidentes à J_e (si J_e est*

incident à une seule face, e^* sera une boucle). Dans le cas orienté, si $e = (v, w)$, nous orienterons $e^* = (F_1, F_2)$ de telle sorte que F_1 soit la face «sur la droite» quand on parcourt J_e de $\psi(v)$ vers $\psi(w)$.

Notons que G^* est bien planaire. Il existe donc une représentation plane $\left(\psi^*, (J_{e^*})_{e^* \in E(G^*)}\right)$ de G^* qui vérifie $\psi^*(F) \in F$ pour toute face F de Φ, $|J_{e^*} \cap J_e| = 1$ pour tout $e \in E(G)$ et

$$J_{e^*} \cap \left(\{\psi(v) : v \in V(G)\} \cup \bigcup_{f \in E(G) \setminus \{e\}} J_f \right) = \emptyset.$$

Une telle représentation est appelée **représentation standard** de G^*.

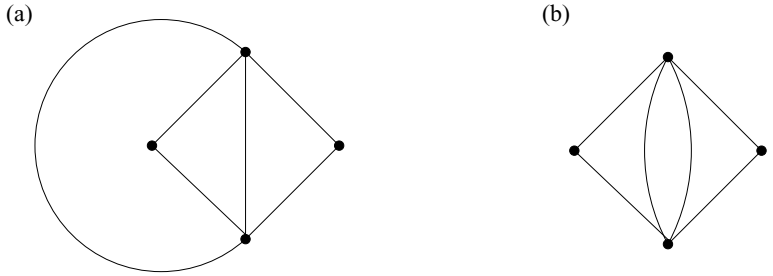


Figure 2.8.

Le dual planaire d'un graphe dépend de sa représentation plane : considérons deux représentations planes du graphe de la figure 2.8. Les deux graphes planaires duaux ne sont pas isomorphes, puisque le second a un sommet de degré quatre correspondant à la face externe tandis que le premier est 3-régulier.

Proposition 2.42. *Soit G un graphe non orienté planaire connexe et une représentation plane fixée de ce graphe. Si G^* est son graphe planaire dual dans une représentation standard, $(G^*)^* = G$.*

Preuve. Soit $\left(\psi, (J_e)_{e \in E(G)}\right)$ une représentation plane fixée de G et soit une représentation standard de G^* : $\left(\psi^*, (J_{e^*})_{e^* \in E(G^*)}\right)$. Soit F une face de G^*. Le bord de F contient J_{e^*} pour au moins une arête e^*, et F contient une extrémité $\psi(v)$ v de e. Donc toute face de G^* contient au moins un sommet de G.

En appliquant la formule d'Euler (théorème 2.32) à G^* et à G, on voit que le nombre de faces de G^* est $|E(G^*)| - |V(G^*)| + 2 = |E(G)| - (|E(G)| - |V(G)| + 2) + 2 = |V(G)|$. Donc chaque face de G^* contient exactement un sommet de G. Nous concluons de cela que le dual planaire de G^* est isomorphe à G. □

L'hypothèse que G est connexe est essentielle ici : notons que G^* est toujours connexe, même si G ne l'est pas.

Théorème 2.43. *Soit G un graphe non orienté planaire connexe avec une représentation plane arbitraire. L'ensemble des arêtes d'un cycle de G correspond à une coupe minimale de G^*, et toute coupe minimale de G correspond à l'ensemble des arêtes d'un cycle de G^*.*

Preuve. Soit $\Phi = (\psi, (J_e)_{e \in E(G)})$ une représentation plane fixée de G. Soit C un cycle de G. Par le théorème 2.30, $\mathbb{R}^2 \setminus \bigcup_{e \in E(C)} J_e$ se sépare en exactement deux régions connexes. Soient A et B l'ensemble des faces de Φ qui appartiennent respectivement à la région interne et à la région externe. $V(G^*) = A \dot\cup B$ et $E_{G^*}(A, B) = \{e^* : e \in E(C)\}$. Puisque A et B sont des ensembles connexes dans G^*, c'est bien une coupe minimale.

Inversement, soit $\delta_G(A)$ une coupe minimale de G. Soit $\Phi^* = (\psi^*, (J_e)_{e \in E(G^*)})$ une représentation plane standard de G^*. Soit $a \in A$ et $b \in V(G) \setminus A$. Observons qu'il n'existe pas de courbe polygonale dans

$$R := \mathbb{R}^2 \setminus \left(\{\psi^*(v) : v \in V(G^*)\} \cup \bigcup_{e \in \delta_G(A)} J_{e^*} \right)$$

qui connecte $\psi(a)$ et $\psi(b)$: la suite des faces de G^* traversées par une telle courbe polygonale induirait une chaîne dans G allant de a à b et n'utilisant aucune arête de $\delta_G(A)$.

Donc R consiste en au moins deux régions connexes et le bord de chaque région contient manifestement un cycle. Donc $F := \{e^* : e \in \delta_G(A)\}$ contient l'ensemble des arêtes d'un cycle C de G^*. Nous savons que $\{e^* : e \in E(C)\} \subseteq \{e^* : e \in F\} = \delta_G(A)$ et, en se référant à la première partie de la démonstration $\{e^* : e \in E(C)\}$ est une coupe minimale de $(G^*)^* = G$ (voir proposition 2.42). Donc $\{e^* : e \in E(C)\} = \delta_G(A)$. □

En particulier, e^* est une boucle d'un graphe si et seulement si e est un pont, et inversement. Pour les graphes orientés nous avons le résultat suivant :

Corollaire 2.44. *Soit G un graphe planaire connexe orienté avec une représentation plane fixée. L'ensemble des arcs d'un circuit de G correspond à une coupe minimale orientée de G^*, et inversement.* □

Une autre conséquence intéressante du théorème 2.43 est :

Corollaire 2.45. *Soit G un graphe connexe non orienté ayant une représentation plane. G est biparti si et seulement si G^* est eulérien, et G est eulérien si et seulement si G^* est biparti.*

Preuve. Observons qu'un graphe connexe est eulérien si et seulement si toute coupe minimale a une cardinalité paire.

Par le théorème 2.43, G est biparti si G^* est eulérien, et G est eulérien si G^* est biparti. Par la proposition 2.42, l'inverse est vrai. □

Un **dual abstrait** de G est un graphe G' pour lequel il existe une bijection $\chi : E(G) \to E(G')$ telle que F est l'ensemble des arêtes d'un cycle si et seulement si $\chi(F)$ est une coupe minimale de G', et inversement. Le théorème 2.43 montre que le graphe planaire dual est également un dual abstrait. L'inverse n'est pas vrai. Cependant, Whitney [1933] a montré qu'un graphe a un dual abstrait si et seulement s'il est planaire (voir exercice 34). Nous reviendrons à cette relation de dualité quand nous étudierons les matroïdes au paragraphe 13.3.

Exercices

1. Soit G un graphe simple non orienté sur n sommets isomorphe à son complément. Montrer que $n \bmod 4 \in \{0, 1\}$.
2. Montrer que tout graphe simple non orienté G avec $|\delta(v)| \geq \frac{1}{2}|V(G)|$ pour tout $v \in V(G)$ est hamiltonien.
   Indication : construire une plus longue chaîne et étudier ses extrémités.
   (Dirac [1952])
3. Montrer que tout graphe simple non orienté G avec $|E(G)| > \binom{|V(G)|-1}{2}$ est connexe.
4. Soit G un graphe simple non orienté. Montrer que G ou son complément est connexe.
5. Montrer que tout graphe simple non orienté avec au moins deux sommets contient deux sommets de même degré. Montrer que tout arbre avec au moins deux sommets contient au moins deux feuilles.
6. Soit G un graphe connexe non orienté et soit $(V(G), F)$ une forêt de G. Montrer qu'il existe un arbre couvrant $(V(G), T)$ avec $F \subseteq T \subseteq E(G)$.
7. Soient (V, F_1) et (V, F_2) deux forêts avec $|F_1| < |F_2|$. Montrer qu'il existe une arête $e \in F_2 \setminus F_1$ telle que $(V, F_1 \cup \{e\})$ soit une forêt.
8. Montrer que toute coupe dans un graphe non orienté est l'union disjointe de coupes minimales.
9. Soient G un graphe non orienté, C un cycle et D une coupe. Montrer que $|E(C) \cap D|$ est pair.
10. Montrer que tout graphe non orienté a une coupe contenant au moins la moitié des arêtes.
11. Soit $(U, \mathcal{F})$ un système d'ensembles sans croisements avec $|U| \geq 2$. Montrer que $\mathcal{F}$ contient au plus $4|U| - 4$ éléments distincts.
12. Soit G un graphe connexe non orienté. Montrer qu'il existe une orientation G' de G et une arborescence couvrante T de G' telle que l'ensemble des cycles fondamentaux par rapport à T soit précisément un ensemble de circuits de G'.
    Indication : regarder un arbre-DFS.
    (Camion [1968], Crestin [1969])

13. Décrire un algorithme linéaire pour le problème suivant : soit une liste d'adjacence d'un graphe G ; calculer une liste d'adjacence du sous-graphe simple maximal de G. On supposera que les arêtes parallèles ne sont pas listées consécutivement dans l'input.

14. Soit G un graphe non orienté ; montrer qu'il existe un algorithme linéaire pour trouver un cycle ou conclure qu'aucun cycle n'existe.

15. Soit G un graphe connexe non orienté avec $s \in V(G)$ et soit T un arbre-DFS résultant de l'exécution de la procédure DFS sur (G, s). s est appelée la racine de T. x est un **prédécesseur** de y dans T si x est sur l'unique chaîne de s à y dans T. x est un **prédécesseur direct (ou parent)** de y si l'arête (x, y) appartient à la chaîne de s à y dans T. y est un **successeur** (resp. un **successeur direct (ou enfant)**) de x si x est un prédécesseur (resp. un prédécesseur direct) de y. Notons qu'avec cette définition chaque sommet est un successeur (et un prédécesseur) de lui-même. Chaque sommet à l'exception de s a exactement un prédécesseur direct. Montrer que :

    (a) Pour toute arête $(v, w) \in E(G)$, v est un prédécesseur ou un successeur de w dans T.

    (b) Un sommet v est un sommet d'articulation de G si et seulement si

    - soit $v = s$ et $|\delta_T(v)| > 1$,
    - ou $v \neq s$ et il n'existe pas de successeur direct w de v tel qu'aucune arête dans G ne connecte un prédécesseur v (en excluant v) à un successeur de w.

∗ 16. Utiliser l'exercice 15 pour élaborer un algorithme linéaire qui trouve les blocs d'un graphe non orienté[3]. Il sera utile de calculer les nombres

$$\alpha(x) := \min\{f(w) : w = x \text{ ou } (w, y) \in E(G) \setminus T \text{ pour un successeur } y \text{ de } x\}$$

récursivement durant le déroulement de DFS. Ici (R, T) est l'arbre-DFS (s étant la racine), et les valeurs f représentent l'ordre selon lequel les sommets sont ajoutés à R (voir l'ALGORITHME DE BALAYAGE DE GRAPHE). Si pour un sommet $x \in R \setminus \{s\}$, $\alpha(x) \geq f(w)$, w étant le prédécesseur direct de x, alors w est soit la racine, soit un sommet d'articulation.

17. Montrer :

    (a) Un graphe non orienté est 2-arête-connexe si et seulement s'il a au moins deux sommets et une décomposition en oreilles.

    (b) Un graphe orienté est fortement connexe si et seulement s'il a une décomposition en oreilles.

    (c) Les arêtes d'un graphe non orienté G avec au moins deux sommets peuvent être orientées de telle sorte que le graphe orienté résultant est fortement connexe si et seulement si G est 2-arête-connexe.
    (Robbins [1939])

[3] Dans tout l'ouvrage, les astérisques signalent les exercices les plus difficiles (*ndt*).

18. Un tournoi est un graphe orienté dont le graphe non orienté associé est un graphe simple complet. Montrer que tout tournoi possède un chemin hamiltonien (Rédei [1934]). Montrer qu'un tournoi fortement connexe est hamiltonien (Camion [1959]).

19. Montrer que si un graphe simple non orienté connexe est eulérien, son *line graph* est hamiltonien. Le contraire est-il vrai ?

20. Montrer que tout graphe biparti connexe a une bipartition unique. Montrer que tout graphe non orienté non biparti contient un cycle impair comme sous-graphe induit.

21. Montrer qu'un graphe orienté fortement connexe contenant un cycle impair contient un circuit impair.

* 22. Soit G un graphe non orienté. Une **décomposition en arbre** de G est une paire (T, φ), où T est un arbre et $\varphi : V(T) \to 2^{V(G)}$ vérifie les conditions suivantes :

- pour tout $e \in E(G)$ il existe $t \in V(T)$ avec $e \subseteq \varphi(t)$;
- pour tout $v \in V(G)$ l'ensemble $\{t \in V(T) : v \in \varphi(t)\}$ est connexe dans T.

La largeur de (T, φ) est $\max_{t \in V(T)} |\varphi(t)| - 1$. La **largeur d'arbre** d'un graphe G est la largeur minimum d'une décomposition en arbre de G. Cette notion est due à Robertson et Seymour [1986].

Montrer que les graphes simples de largeur d'arbre au plus 1 sont les forêts. Montrer que les conditions suivantes sont équivalentes pour tout graphe non orienté G :

(a) G a une largeur d'arbre au plus 2.

(b) G ne contient pas de mineur K_4.

(c) G peut être obtenu à partir du graphe vide en ajoutant successivement des ponts et en doublant et en subdivisant des arêtes. (Doubler une arête $e = (v, w) \in E(G)$ signifie ajouter une nouvelle arête d'extrémités v et w ; subdiviser une arête $e = (v, w) \in E(G)$ signifie ajouter un sommet x et remplacer e par deux arêtes $(v, x), (x, w)$.)

Note : ces graphes, à cause de la construction (c), sont appelés série-parallèles.

23. Montrer que si un graphe G a une représentation plane où les arêtes sont représentées par des courbes de Jordan arbitraires, il a aussi une représentation plane avec des courbes polygonales.

24. Soit G un graphe 2-connexe avec une représentation plane. Montrer que l'ensemble des cycles bordant les faces finies constituent une base des cycles de G.

25. Peut-on généraliser la formule d'Euler (théorème 2.32) aux graphes non connexes ?

26. Montrer qu'il existe exactement cinq graphes platoniques (correspondant aux solides platoniques ; voir exercice 11 du chapitre 4), c.-à-d. des graphes 3-connexes réguliers planaires dont les faces sont bordées par le même nombre

d'arêtes.
Indication : utiliser la formule d'Euler (théorème 2.32).

27. Déduire de la preuve du théorème de Kuratowski 2.39 :
    (a) Tout graphe simple planaire 3-connexe a une représentation plane où chaque arête est représentée par un segment de droite et chaque face, à l'exception de la face externe, est convexe.
    (b) Il existe un algorithme polynomial pour tester si un graphe est planaire.

∗ 28. Soit un graphe G et soit une arête $e = (v, w) \in E(G)$; nous dirons que H se déduit de G par subdivision de e si $V(H) = V(G) \dot\cup \{x\}$ et $E(H) = (E(G) \setminus \{e\}) \cup \{(v, x), (x, w)\}$. Un graphe résultant de G par subdivisions successives d'arêtes est appelé une **subdivision** de G.
    (a) Manifestement, si H contient une subdivision de G, G est un mineur de H. Montrer que le contraire n'est pas vrai.
    (b) Montrer qu'un graphe contenant $K_{3,3}$ ou K_5 comme mineur contient aussi une subdivision de $K_{3,3}$ ou de K_5.
    Indication : étudier ce qui se passe quand on contracte une arête.
    (c) Conclure qu'un graphe est planaire si et seulement si aucun de ses sous-graphes n'est une subdivision de $K_{3,3}$ ou K_5.

    (Kuratowski [1930])

29. Montrer que les conditions suivantes sont équivalentes :
    (a) Pour toute famille infinie de graphes $G_1, G_2, \ldots$ il existe deux indices $i < j$ tels que G_i soit un mineur de G_j.
    (b) Soit $\mathcal{G}$ une classe de graphes tels que pour chaque $G \in \mathcal{G}$ et pour chaque mineur H de G, $H \in \mathcal{G}$ (c.-à-d. l'appartenance à $\mathcal{G}$ soit une propriété de graphes héréditaire). Alors il existe un ensemble fini $\mathcal{X}$ de graphes tel que $\mathcal{G}$ consiste en tous les graphes qui ne contiennent pas un élément de $\mathcal{X}$ comme mineur.

    Note : ces équivalences ont été montrées par Robertson et Seymour [2004] ; elles constituent un résultat essentiel de leur série d'articles sur les mineurs dans les graphes. Le théorème 2.39 de l'exercice 22 sont des illustrations de caractérisations par mineurs exclus comme dans (b).

30. Soit G un graphe planaire avec une représentation plane Φ, et soit C un cycle de G bordant une face de Φ. Montrer qu'il existe une autre représentation Φ' G où C borde la face extérieure.

31. (a) Soit G un graphe connexe avec une représentation plane arbitraire, et soit G^* le graphe dual planaire avec une représentation plane standard. Montrer que $(G^*)^*$ se déduit de G en appliquant successivement les opérations suivantes jusqu'à ce que le graphe devienne connexe : choisir deux sommets x et y qui sont dans des composantes différentes et qui sont adjacents à la même face ; contracter (x, y).

(b) Généraliser le corollaire 2.45 à des graphes planaires arbitraires.
Indication : utiliser (a) et le théorème 2.26.

32. Soit G un graphe connexe orienté avec une représentation plane, et soit G^* son dual planaire avec une représentation plane standard. Comment G et $(G^*)^*$ sont-ils reliés ?

33. Montrer que si un graphe planaire orienté est sans circuit (fortement connexe), alors son dual planaire est fortement connexe (sans circuit). Que peut-on dire de l'inverse ?

34. (a) Montrer que si G a un dual abstrait et si H est un mineur de G, alors H a aussi un dual abstrait.

$*$ (b) Montrer que ni K_5 ni $K_{3,3}$ n'ont de dual abstrait.

(c) Conclure qu'un graphe est planaire si et seulement s'il a un dual abstrait.
(Whitney [1933])

Références

Littérature générale :

Berge, C. [1985] : Graphs. Second Edition. Elsevier, Amsterdam 1985

Bollobás, B. [1998] : Modern Graph Theory. Springer, New York 1998

Bondy, J.A. [1995] : Basic graph theory : paths and circuits. In : Handbook of Combinatorics ; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam 1995

Bondy, J.A., Murty, U.S.R. [2008] : Graph Theory. Springer, New York 2008

Diestel, R. [2005] : Graph Theory. Third Edition. Springer, New York 2005

Wilson, R.J. [1996] : Introduction to Graph Theory. Fourth Edition. Addison-Wesley, Reading 1996

Références citées :

Aoshima, K., Iri, M. [1977] : Comments on F. Hadlock's paper : finding a maximum coupe of a Planar graph in polynomial time. SIAM Journal on Computing 6 (1977), 86–87

Camion, P. [1959] : Chemins et circuits hamiltoniens des graphes complets. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris) 249 (1959), 2151–2152

Camion, P. [1968] : Modulaires unimodulaires. Journal of Combinatorial Theory A 4 (1968), 301–362

Dirac, G.A. [1952] : Some theorems on abstract graphs. Proceedings of the London Mathematical Society 2 (1952), 69–81

Edmonds, J., Giles, R. [1977] : A min-max relation for submodular functions on graphs. In : Studies in Integer Programming ; Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, eds.), North-Holland, Amsterdam 1977, pp. 185–204

Euler, L. [1736] : Solutio Problematis ad Geometriam Situs Pertinentis. Commentarii Academiae Petropolitanae 8 (1736), 128–140

Euler, L. [1758] : Demonstratio nonnullarum insignium proprietatum quibus solida hedris planis inclusa sunt praedita. Novi Commentarii Academiae Petropolitanae 4 (1758), 140–160

Hierholzer, C. [1873] : Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen 6 (1873), 30–32

Hopcroft, J.E., Tarjan, R.E. [1974] : Efficient planarity testing. Journal of the ACM 21 (1974), 549–568

Kahn, A.B. [1962] : Topological sorting of large networks. Communications of the ACM 5 (1962), 558–562

Knuth, D.E. [1968] : The Art of Computer Programming ; Vol. 1 ; Fundamental Algorithms. Addison-Wesley, Reading 1968 (third edition : 1997)

König, D. [1916] : Über Graphen und Ihre Anwendung auf Determinantentheorie und Mengenlehre. Mathematische Annalen 77 (1916), 453–465

König, D. [1936] : Theorie der endlichen und unendlichen Graphen. Chelsea Publishing Co., Leipzig 1936, reprint New York 1950

Kuratowski, K. [1930] : Sur le problème des courbes gauches en topologie. Fundamenta Mathematicae 15 (1930), 271–283

Legendre, A.M. [1794] : Éléments de Géométrie. Firmin Didot, Paris 1794

Minty, G.J. [1960] : Monotone networks. Proceedings of the Royal Society of London A 257 (1960), 194–212

Moore, E.F. [1959] : The shortest path through a maze. Proceedings of the International Symposium on the Theory of Switching ; Part II. Harvard University Press 1959, pp. 285–292

Rédei, L. [1934] : Ein kombinatorischer Satz. Acta Litt. Szeged 7 (1934), 39–43

Robbins, H.E. [1939] : A theorem on graphs with an application to a problem of traffic control. American Mathematical Monthly 46 (1939), 281–283

Robertson, N., Seymour, P.D. [1986] : Graph minors II : algorithmic aspects of arbre-width. Journal of Algorithms 7 (1986), 309–322

Robertson, N., Seymour, P.D. [2004] : Graph minors XX : Wagner's conjecture. Journal of Combinatorial Theory B 92 (2004), 325–357

Tarjan, R.E. [1972] : Depth first search and linear graph algorithms. SIAM Journal on Computing 1 (1972), 146–160

Thomassen, C. [1980] : Planarity and duality of finite and infinite graphs. Journal of Combinatorial Theory B 29 (1980), 244–271

Thomassen, C. [1981] : Kuratowski's theorem. Journal of Graph Theory 5 (1981), 225–241

Tutte, W.T. [1961] : A theory of 3-connected graphs. Konink. Nederl. Akad. Wetensch. Proc. A 64 (1961), 441–455

Wagner, K. [1937] : Über eine Eigenschaft der ebenen Komplexe. Mathematische Annalen 114 (1937), 570–590

Whitney, H. [1932] : Non-separable and planar graphs. Transactions of the American Mathematical Society 34 (1932), 339–362

Whitney, H. [1933] : Planar graphs. Fundamenta Mathematicae 21 (1933), 73–84

Chapitre 3

Programmation linéaire

Dans ce chapitre, nous passons en revue les aspects principaux de la PROGRAMMATION LINÉAIRE. Bien que se suffisant à elle-même, cette présentation nécessite cependant une connaissance préalable du sujet, que le lecteur non averti pourra trouver dans les livres de référence mentionnés en fin de chapitre.

Le problème peut se formuler comme suit :

PROGRAMMATION LINÉAIRE	
Instance	Une matrice $A \in \mathbb{R}^{m \times n}$ et deux vecteurs colonnes $b \in \mathbb{R}^m, c \in \mathbb{R}^n$.
Tâche	Trouver un vecteur colonne $x \in \mathbb{R}^n$ tel que $Ax \leq b$ et $c^\top x$ soit maximum, décider si $\{x \in \mathbb{R}^n : Ax \leq b\}$ est vide, ou décider que pour tout $\alpha \in \mathbb{R}$ il existe $x \in \mathbb{R}^n$ tel que $Ax \leq b$ et $c^\top x > \alpha$.

$c^\top x$ est ici le produit scalaire. Si x et y sont deux vecteurs de même dimension, $x \leq y$ signifie que l'inégalité est vérifiée pour chaque composante de ces vecteurs. Si aucune taille n'est mentionnée, les vecteurs et les matrices seront supposés être compatibles en taille. Nous omettrons souvent le signe de transposition pour un vecteur colonne et nous écrirons cx pour le produit scalaire. 0 sera aussi bien le nombre zéro que le vecteur ou la matrice de composantes nulles.

Un **programme linéaire, PL** en abrégé, est une instance du problème précédent. Un PL sera souvent écrit $\max\{cx : Ax \leq b\}$. Une **solution réalisable** d'un PL $\max\{cx : Ax \leq b\}$ est un vecteur x qui vérifie $Ax \leq b$. Une solution réalisable atteignant le maximum est appelée **solution optimale**.

Comme la formulation le suggère, il y a deux possibilités pour un PL qui n'a pas de solution : soit le problème est **non réalisable** ($P := \{x \in \mathbb{R}^n : Ax \leq b\} = \emptyset$), soit il est **non borné** (pour tout $\alpha \in \mathbb{R}$ il existe $x \in P$ avec $cx > \alpha$). Si un PL n'est ni non réalisable ni non borné il a une solution optimale :

Proposition 3.1. *Soit $P = \{x \in \mathbb{R}^n : Ax \leq b\} \neq \emptyset$ et $c \in \mathbb{R}^n$ avec $\delta := \sup\{c^\top x : x \in P\} < \infty$. Alors il existe un vecteur $z \in P$ avec $c^\top z = \delta$.*

Preuve. Soit U une matrice dont les colonnes forment une base orthonormée du noyau de A : $U^\top U = I$, $AU = 0$, et $\mathrm{rang}(A') = n$ où $A' := \begin{pmatrix} A \\ U^\top \end{pmatrix}$. Soit $b' := \begin{pmatrix} b \\ 0 \end{pmatrix}$.

Montrons que pour chaque $y \in P$ il existe un sous-système $A''x \leq b''$ de $A'x \leq b'$ vérifiant : A'' est non singulière, $y' := (A'')^{-1}b'' \in P$, et $c^\top y' \geq c^\top y$. Comme le nombre de ces sous-systèmes est fini, la valeur δ sera atteinte pour un de ces y', (c.-à-d. $c^\top y' = \delta$), ce qui prouvera le résultat.

Soit $y \in P$, et notons par $k(y)$ le rang de A'', $A''x \leq b''$ étant le sous-système maximal de $A'x \leq b'$ tel que $A''y = b''$. Montrons que, si $k(y) < n$, on peut trouver $y' \in P$ avec $c^\top y' \geq c^\top y$ et $k(y') > k(y)$. Après au plus n itérations nous aurons donc un vecteur y' tel que $k(y') = n$, comme souhaité.

Si $U^\top y \neq 0$, posons $y' := y - UU^\top y$. Puisque $y + \lambda UU^\top c \in P$ pour tout $\lambda \in \mathbb{R}$, $\sup\{c^\top(y + \lambda UU^\top c) : \lambda \in \mathbb{R}\} \leq \delta < \infty$ et donc $c^\top U = 0$ et $c^\top y' = c^\top y$. De plus, $Ay' = Ay - AUU^\top y = Ay$ et $U^\top y' = U^\top y - U^\top UU^\top y = 0$.

Supposons donc que $U^\top y = 0$. Soit $v \neq 0$ vérifiant $A''v = 0$. Notons par $a_i x \leq \beta_i$ la i-ième ligne de $Ax \leq b$. Soit $\mu := \min\left\{\frac{\beta_i - a_i y}{a_i v} : a_i v > 0\right\}$ et $\kappa := \max\left\{\frac{\beta_i - a_i y}{a_i v} : a_i v < 0\right\}$, où $\min\emptyset = \infty$ et $\max\emptyset = -\infty$. $\kappa \leq 0 \leq \mu$, et au moins un des deux nombres κ, μ est fini.

Si $\lambda \in \mathbb{R}$ avec $\kappa \leq \lambda \leq \mu$ $A''(y + \lambda v) = A''y + \lambda A''v = A''y = b''$ et $A(y + \lambda v) = Ay + \lambda Av \leq b$, i.e. $y + \lambda v \in P$. Donc, comme $\sup\{c^\top x : x \in P\} < \infty$, $\mu < \infty$ si $c^\top v > 0$ et $\kappa > -\infty$ si $c^\top v < 0$.

De plus, si $c^\top v \geq 0$ et $\mu < \infty$, alors $a_i(y + \mu v) = \beta_i$ pour un indice i. De même, si $c^\top v \leq 0$ et $\kappa > -\infty$, alors $a_i(y + \kappa v) = \beta_i$ pour un indice i.

Dans chacun des deux cas nous avons trouvé un vecteur $y' \in P$ tel que $c^\top y' \geq c^\top y$ et $k(y') \geq k(y) + 1$. □

Cela justifie la notation $\max\{c^\top x : Ax \leq b\}$ à la place de $\sup\{c^\top x : Ax \leq b\}$.

De nombreux problèmes d'optimisation combinatoire peuvent être formulés comme des PL. Pour cela, il nous faut représenter les solutions réalisables comme des vecteurs d'un espace $\mathbb{R}^n$. Nous montrerons, au paragraphe 3.5, comment optimiser une fonction objectif linéaire sur un ensemble fini S de vecteurs par résolution d'un PL. Bien que l'ensemble des solutions de ce PL contienne non seulement les éléments de S, mais aussi leurs combinaisons convexes, on peut montrer qu'il existe toujours une solution optimale qui est dans S.

Nous rappellerons quelques notions de base relatives aux polyèdres, ensembles $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ des solutions réalisables au paragraphe 3.1. Nous présenterons l'ALGORITHME DU SIMPLEXE aux paragraphes 3.2 et 3.3 ; nous utiliserons cet algorithme pour montrer le théorème de dualité ainsi que des résultats connexes (paragraphe 3.4). La dualité est une notion essentielle qui apparaît de manière explicite ou implicite dans presque tous les aspects de l'optimisation combinatoire ; nous utiliserons souvent les résultats des paragraphes 3.4 et 3.5.

3.1 Polyèdres

La PROGRAMMATION LINÉAIRE s'intéresse à la maximisation ou à la minimisation d'une fonction objectif linéaire définie sur un ensemble fini de variables qui vérifient un nombre fini d'inégalités linéaires. L'ensemble des solutions réalisables est l'intersection d'un nombre fini de demi-espaces. Un tel ensemble est un polyèdre :

Définition 3.2. *Un* **polyèdre** *de $\mathbb{R}^n$ est un ensemble $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ où $A \in \mathbb{R}^{m \times n}$ est une matrice et $b \in \mathbb{R}^m$ est un vecteur. Si A et b sont rationnels, P est un* **polyèdre rationnel**. *Un polyèdre borné est aussi appelé un* **polytope**.

Le rang de la matrice A sera noté $\operatorname{rang}(A)$. *La* **dimension**, $\dim X$, *d'un ensemble non vide $X \subseteq \mathbb{R}^n$ est égale à*

$$n - \max\{\operatorname{rang}(A) : A \text{ est une matrice } n \times n \text{ avec } Ax = Ay \text{ pour tout } x, y \in X\}.$$

Un polyèdre $P \subseteq \mathbb{R}^n$ est de **pleine dimension** *si* $\dim P = n$.

Notons qu'un polyèdre est de pleine dimension si et seulement si son intérieur est non vide. On supposera dans l'essentiel de ce chapitre que l'espace vectoriel de référence est réel ou rationnel. Introduisons la terminologie suivante :

Définition 3.3. *Soit $P := \{x : Ax \leq b\}$ un polyèdre non vide. Si c est un vecteur différent de zéro tel que $\delta := \max\{cx : x \in P\}$ est fini, alors $\{x : cx = \delta\}$ sera un* **hyperplan support** *de P. Une* **face** *de P est P lui-même ou l'intersection de P avec un hyperplan support de P. Un point x tel que $\{x\}$ est une face sera un* **sommet** *de P, et aussi une* **solution de base** *du système $Ax \leq b$.*

Proposition 3.4. *Soit $P = \{x : Ax \leq b\}$ un polyèdre et $F \subseteq P$. Les conditions suivantes sont équivalentes :*

(a) *F est une face de P.*

(b) *Il existe un vecteur c tel que $\delta := \max\{cx : x \in P\}$ est fini et $F = \{x \in P : cx = \delta\}$.*

(c) *$F = \{x \in P : A'x = b'\} \neq \emptyset$ pour un sous-système $A'x \leq b'$ de $Ax \leq b$.*

Preuve. Les conditions (a) et (b) sont clairement équivalentes.

(c)⇒(b) : si $F = \{x \in P : A'x = b'\}$ est non vide, soit c le vecteur somme des lignes de A', et soit δ la somme des composantes de b'. Alors $cx \leq \delta$ pour tout $x \in P$ et $F = \{x \in P : cx = \delta\}$.

(b)⇒(c) : soit c est un vecteur tel que $\delta := \max\{cx : x \in P\}$ est fini et soit $F = \{x \in P : cx = \delta\}$. Soit $A'x \leq b'$ le sous-système maximal de $Ax \leq b$ tel que $A'x = b'$ pour tout $x \in F$. Soit $A''x \leq b''$ l'autre partie du système $Ax \leq b$.

Observons que pour chaque inégalité $a_i''x \leq \beta_i''$ de $A''x \leq b''$ $(i = 1, \ldots, k)$ il existe un point $x_i \in F$ tel que $a_i''x_i < \beta_i''$. Soit $x^* := \frac{1}{k}\sum_{i=1}^{k} x_i$ le centre de gravité de ces points (si $k = 0$, choisissons un $x^* \in F$ arbitraire) ; alors, $x^* \in F$ et $a_i''x^* < \beta_i''$ pour tout i.

Montrons que nous ne pouvons avoir $A'y = b'$ si $y \in P \setminus F$. Soit $y \in P \setminus F$; on a alors $cy < \delta$. Soit $z := x^* + \epsilon(x^* - y)$ pour $\epsilon > 0$ suffisamment petit ; en particulier choisissons ϵ plus petit que $\frac{\beta_i'' - a_i'' x^*}{a_i''(x^* - y)}$ pour tout $i \in \{1, \ldots, k\}$ tel que $a_i'' x^* > a_i'' y$.

Comme $cz > \delta$, $z \notin P$. Il existe alors une inégalité $ax \leq \beta$ de $Ax \leq b$ telle que $az > \beta$. Donc $ax^* > ay$. L'inégalité $ax \leq \beta$ ne peut pas appartenir à $A''x \leq b''$, sinon nous aurions $az = ax^* + \epsilon a(x^* - y) < ax^* + \frac{\beta - ax^*}{a(x^* - y)} a(x^* - y) = \beta$ (par le choix de ϵ). Cette inégalité $ax \leq \beta$ appartient donc à $A'x \leq b'$. Puisque $ay = a(x^* + \frac{1}{\epsilon}(x^* - z)) < \beta$, l'implication est démontrée. □

Notons un corollaire évident mais important :

Corollaire 3.5. *Si $\max\{cx : x \in P\}$ est borné pour P non vide et pour le vecteur c, alors l'ensemble des points qui atteignent le maximum est une face de P.* □

La relation «est une face de» est transitive :

Corollaire 3.6. *Soit P un polyèdre et soit F une face de P. Alors F est un polyèdre. De plus, un ensemble $F' \subseteq F$ est une face de P si et seulement si c'est une face de F.* □

Les faces maximales distinctes de P sont particulièrement importantes :

Définition 3.7. *Soit P un polyèdre. Une* **facette** *de P est une face maximale distincte de P. Une inégalité $cx \leq \delta$* **induit une facette** *pour P si $cx \leq \delta$ pour tout $x \in P$ et $\{x \in P : cx = \delta\}$ est une facette de P.*

Proposition 3.8. *Soit $P \subseteq \{x \in \mathbb{R}^n : Ax = b\}$ un polyèdre non vide de dimension $n - \operatorname{rang}(A)$. Soit $A'x \leq b'$ un système minimal d'inégalités tel que $P = \{x : Ax = b, A'x \leq b'\}$. Alors chaque inégalité de $A'x \leq b'$ induit une facette pour P, et chaque facette de P est induite par une inégalité de $A'x \leq b'$.*

Preuve. Si $P = \{x \in \mathbb{R}^n : Ax = b\}$, il n'existe aucune facette et la proposition est évidente. Si $P = \{x : Ax = b, A'x \leq b'\}$, $A'x \leq b'$ étant un système minimal d'inégalités, soit $a'x \leq \beta'$ une de ces inégalités et soit $A''x \leq b''$ l'autre partie du système $A'x \leq b'$. Soit y un vecteur tel que $Ay = b$, $A''y \leq b''$ et $a'y > \beta'$ (un tel vecteur y existe puisque l'inégalité $a'x \leq \beta'$ n'est pas redondante). Soit $x \in P$ tel que $a'x < \beta'$ (un tel vecteur existe puisque $\dim P = n - \operatorname{rang}(A)$).

Soit $z := x + \frac{\beta' - a'x}{a'y - a'x}(y - x) : a'z = \beta'$ et, puisque $0 < \frac{\beta' - a'x}{a'y - a'x} < 1$, $z \in P$. Donc $F := \{x \in P : a'x = \beta'\} \neq 0$ et $F \neq P$ (puisque $x \in P \setminus F$). Donc F est une facette de P. Par la proposition 3.4 toute facette est induite par une inégalité de $A'x \leq b'$. □

En dehors des facettes, les autres faces importantes sont les faces minimales (c.-à-d. les faces ne contenant aucune autre face) :

Proposition 3.9. (Hoffman et Kruskal [1956]) *Soit $P = \{x : Ax \leq b\}$ un polyèdre ; un sous-ensemble non vide $F \subseteq P$ est une face minimale de P si et seulement si $F = \{x : A'x = b'\}$ où $A'x \leq b'$ est un sous-système de $Ax \leq b$.*

Preuve. Si F est une face minimale de P, il existe par la proposition 3.4 un sous-système $A'x \leq b'$ de $Ax \leq b$ tel que $F = \{x \in P : A'x = b'\}$. On peut supposer que $A'x \leq b'$ est maximal. Soit $A''x \leq b''$ un sous-système minimal de $Ax \leq b$ tel que $F = \{x : A'x = b',\ A''x \leq b''\}$. Montrons que $A''x \leq b''$ ne contient aucune inégalité.

Supposons, au contraire, que $a''x \leq \beta''$ soit une inégalité de $A''x \leq b''$. Cette inégalité n'étant pas redondante dans la description de F, $F' := \{x : A'x = b',\ A''x \leq b'',\ a''x = \beta''\}$ est une facette de F (proposition 3.8). Par le corollaire 3.6 F' est aussi une face de P, ce qui contredit l'hypothèse de minimalité de F.

Soit $\emptyset \neq F = \{x : A'x = b'\} \subseteq P$, $A'x \leq b'$ étant un sous-système de $Ax \leq b$. Il est évident que F ne contient aucune face à part elle-même. Par la proposition 3.4, F est une face de P. Par le corollaire 3.6, F est une face minimale de P. □

Le corollaire 3.5 et la proposition 3.9 impliquent qu'on peut résoudre la PROGRAMMATION LINÉAIRE en temps fini : il suffit de résoudre $A'x = b'$ pour tout sous-système $A'x \leq b'$ de $Ax \leq b$. Une manière plus efficace sera l'ALGORITHME DU SIMPLEXE que nous présenterons dans le paragraphe suivant.

Une autre conséquence de la proposition 3.9 est :

Corollaire 3.10. *Soit $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ un polyèdre ; la dimension de toute face minimale de P est $n - \mathrm{rang}(A)$; les faces minimales des polytopes sont les sommets.* □

C'est pour cette raison que les polyèdres $\{x \in \mathbb{R}^n : Ax \leq b\}$ pour lesquels $\mathrm{rang}(A) = n$ sont dits **pointés** : leurs faces minimales sont des points.

Terminons ce paragraphe avec quelques remarques sur les cônes polyédraux.

Définition 3.11. *Un* **cône** *(convexe) est un ensemble $C \subseteq \mathbb{R}^n$ tel que $\lambda x + \mu y \in C$ pour tout $x, y \in C$ et $\lambda, \mu \geq 0$. Un* **cône** *C est* **engendré** *par $x_1, \ldots, x_k$ si $x_1, \ldots, x_k \in C$ et s'il existe des nombres $\lambda_1, \ldots, \lambda_k \geq 0$ tels que $x = \sum_{i=1}^{k} \lambda_i x_i$ pour tout $x \in C$. Un cône est* **finiment engendré** *s'il est engendré par un nombre fini de vecteurs. Un* **cône polyédral** *est un polyèdre du type $\{x : Ax \leq 0\}$.*

Il est immédiat de vérifier que les cônes polyédraux sont des cônes. Nous allons démontrer que les cônes polyédraux sont finiment engendrés. I sera toujours la matrice identité.

Lemme 3.12. (Minkowski [1896]) *Soit $C = \{x \in \mathbb{R}^n : Ax \leq 0\}$ un cône polyédral. Alors C est engendré par un sous-ensemble des solutions des systèmes $My = b'$, où M est une sous-matrice $n \times n$ de $\binom{A}{I}$ non singulière et où $b' = \pm e_j$ (e_j étant un vecteur unité).*

Preuve. Soit A une matrice $m \times n$. Considérons les systèmes $My = b'$ où M est constituée de n lignes linéairement indépendantes de $\binom{A}{I}$ et $b' = \pm e_j$, e_j étant un vecteur unité. Soient $y_1, \ldots, y_t$ l'ensemble des solutions de ces systèmes qui appartiennent à C. Nous allons montrer que C est engendré par $y_1, \ldots, y_t$.

Supposons d'abord que C soit un sous-espace linéaire : $C = \{x : Ax = 0\}$. On a aussi $C = \{x : A'x = 0\}$ où A' consiste en un nombre maximal de lignes linéairement indépendantes de A. Soit I' un sous-ensemble de lignes de I tel que $\binom{A'}{I'}$ soit une matrice carrée non singulière. C est engendré par les solutions de

$$\begin{pmatrix} A' \\ I' \end{pmatrix} x = \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad \text{pour } b = \pm e_j,\ j = 1, \ldots, \dim C.$$

Dans le cas général nous ferons une induction sur la dimension de C. Si C n'est pas un sous-espace linéaire, choisissons une ligne a de A et une sous-matrice A' de A de telle sorte que les lignes de $\binom{A'}{a}$ soient linéairement indépendantes et $\{x : A'x = 0,\ ax \leq 0\} \subseteq C$. Par construction, il existe un indice $s \in \{1, \ldots, t\}$ tel que $A'y_s = 0$ et $ay_s = -1$.

Choisissons arbitrairement $z \in C$. Soient $a_1, \ldots, a_m$ les lignes de A et $\mu := \min\left\{\frac{a_i z}{a_i y_s} : i = 1, \ldots, m,\ a_i y_s < 0\right\}$. Notons que $\mu \geq 0$. Soit k un indice pour lequel le minimum est atteint. Posons $z' := z - \mu y_s$. Par la définition de μ nous avons $a_j z' = a_j z - \frac{a_k z}{a_k y_s} a_j y_s$ pour $j = 1, \ldots, m$, et donc $z' \in C' := \{x \in C : a_k x = 0\}$. C' est un cône dont la dimension est diminuée de 1 par rapport à celle de C (parce que $a_k y_s < 0$ et $y_s \in C$). Par induction, C' est engendré par $y_1, \ldots, y_t$, et $z' = \sum_{i=1}^{t} \lambda_i y_i$ avec $\lambda_1, \ldots, \lambda_t \geq 0$. En posant $\lambda'_s := \lambda_s + \mu$ (observons que $\mu \geq 0$) et $\lambda'_i := \lambda_i$ $(i \neq s)$, nous obtenons $z = z' + \mu y_s = \sum_{i=1}^{t} \lambda'_i y_i$. □

Ainsi tout cône polyédral est finiment engendré. Nous montrerons l'inverse à la fin du paragraphe 3.4.

3.2 Algorithme du simplexe

L'algorithme de PROGRAMMATION LINÉAIRE le plus célèbre et le plus ancien est la méthode du simplexe de Dantzig [1951]. Nous supposerons d'abord que le polyèdre a un sommet, et que ce sommet fait partie de l'input. Plus tard nous lèverons cette restriction.

Si J est un ensemble d'indices de lignes, A_J est la sous-matrice de A indicée sur les lignes de J, et b_J est la restriction de b à J. Nous écrirons d'autre part $a_i := A_{\{i\}}$ et $\beta_i := b_{\{i\}}$.

ALGORITHME DU SIMPLEXE	
Input	Une matrice $A \in \mathbb{R}^{m \times n}$ et deux vecteurs colonnes $b \in \mathbb{R}^m, c \in \mathbb{R}^n$. Un sommet x de $P := \{x \in \mathbb{R}^n : Ax \leq b\}$.
Output	Un sommet x de P réalisant $\max\{cx : x \in P\}$ ou un vecteur $w \in \mathbb{R}^n$ vérifiant $Aw \leq 0$ et $cw > 0$ (c.-à-d. le PL est non borné).

① Choisir un ensemble J de n indices de lignes tel que A_J soit non singulière et soit x la solution du système $A_J x = b_J$.

② Calculer $c\,(A_J)^{-1}$ et ajouter des zéros de telle manière à obtenir un vecteur y avec $c = yA$ dont toutes les composantes en dehors de J valent zéro.
If $y \geq 0$ **then stop**. **Return** x et y.

③ Choisir l'indice minimum i tel que $y_i < 0$.
Soit w la colonne de $-(A_J)^{-1}$ et soit $i \in J$ tels que $A_{J\setminus\{i\}} w = 0$ et $a_i w = -1$.
If $Aw \leq 0$ **then stop**.
Return w.

④ Soit $\lambda := \min\left\{\dfrac{\beta_j - a_j x}{a_j w} : j \in \{1,\ldots,m\},\ a_j w > 0\right\}$,
et soit j le plus petit indice de ligne atteignant ce minimum.

⑤ $J := (J \setminus \{i\}) \cup \{j\}$ et $x := x + \lambda w$.
Go to ②.

L'étape ① est fondée sur la proposition 3.9 et peut être implémentée par ÉLIMINATION DE GAUSS (paragraphe 4.3). Les règles de choix pour i et j dans ③ et ④ (souvent appelées règle du pivot) sont dues à Bland [1977]. Si on choisit arbitrairement i tel que $y_i < 0$ et un des j réalisant le minimum dans ④, l'algorithme pourrait boucler dans certains cas. La règle du pivot de Bland n'est pas la seule qui évite le bouclage ; une autre règle (la règle lexicographique) permet aussi d'éviter de boucler indéfiniment (Dantzig, Orden et Wolfe [1955]). Avant de démontrer la validité de l'ALGORITHME DU SIMPLEXE, faisons l'observation suivante (appelée quelquefois propriété de la «dualité faible») :

Proposition 3.13. *Soient x et y deux solutions réalisables respectives des PLs*

$$\max\{cx : Ax \leq b\} \qquad \textit{et} \tag{3.1}$$
$$\min\{yb : y^\top A = c^\top,\ y \geq 0\}. \tag{3.2}$$

Alors $cx \leq yb$.

Preuve. $cx = (yA)x = y(Ax) \leq yb$. □

Théorème 3.14. (Dantzig [1951], Dantzig, Orden et Wolfe [1955], Bland [1977]) *L'*ALGORITHME DU SIMPLEXE *se termine après au plus $\binom{m}{n}$ itérations. S'il retourne x et y dans ②, ces vecteurs sont les solutions optimales respectives des PL (3.1) et (3.2) et $cx = yb$. S'il retourne w dans ③ $cw > 0$ et le PL (3.1) est non borné.*

Preuve. Montrons que les conditions suivantes sont vérifiées à toute étape de l'algorithme :

(a) $x \in P$.

(b) $A_J x = b_J$.

(c) A_J est non singulière.

(d) $cw > 0$.

(e) $\lambda \geq 0$.

(a) et (b) sont vérifiées initialement. ② et ③ garantissent $cw = yAw = -y_i > 0$. Par ④, $x \in P$ implique que $\lambda \geq 0$. (c) est vraie parce que $A_{J\setminus\{i\}} w = 0$ et $a_j w > 0$. Il reste à montrer qu'après ⑤ (a) et (b) restent vraies.

Montrons que si $x \in P$, alors $x + \lambda w \in P$. Si k est un indice de ligne, nous avons deux cas : si $a_k w \leq 0$, $a_k(x + \lambda w) \leq a_k x \leq \beta_k$ (puisque $\lambda \geq 0$). Sinon $\lambda \leq \frac{\beta_k - a_k x}{a_k w}$ et alors $a_k(x + \lambda w) \leq a_k x + a_k w \frac{\beta_k - a_k x}{a_k w} = \beta_k$. (La valeur λ étant choisie dans ④ la plus grande possible pour que $x + \lambda w \in P$.)

Pour montrer (b), notons qu'après ④ nous avons $A_{J\setminus\{i\}} w = 0$ et $\lambda = \frac{\beta_j - a_j x}{a_j w}$; ainsi $A_{J\setminus\{i\}}(x+\lambda w) = A_{J\setminus\{i\}} x = b_{J\setminus\{i\}}$ et $a_j(x+\lambda w) = a_j x + a_j w \frac{\beta_j - a_j x}{a_j w} = \beta_j$. Donc après ⑤, $A_J x = b_J$ continue à être vérifié.

(a)–(e) sont donc vérifiées à chaque étape. Si l'algorithme retourne x et y en ②, x et y sont solutions réalisables respectives de (3.1) et (3.2). x est un sommet de P par (a), (b) et (c). Puisque les composantes de y valent zéro en dehors de J, on a $cx = yAx = yb$. Cela montre l'optimalité de x et y par la proposition 3.13.

Si l'algorithme s'arrête en in ③, le PL (3.1) est non borné, car dans ce cas $x + \mu w \in P$ pour tout $\mu \geq 0$, et $cw > 0$ par (d).

Montrons finalement que l'algorithme se termine. Soit $J^{(k)}$ l'ensemble J et soit $x^{(k)}$ le vecteur x à l'itération k de l'ALGORITHME DU SIMPLEXE. Si l'algorithme ne s'arrête pas après $\binom{m}{n}$ itérations, il y a deux itérations $k < l$ telles que $J^{(k)} = J^{(l)}$. Par (b) et (c), $x^{(k)} = x^{(l)}$. Par (d) et (e), cx ne décroît jamais et croît strictement si $\lambda > 0$. Donc λ est égal à zéro dans toutes les itérations $k, k+1, \ldots, l-1$, et $x^{(k)} = x^{(k+1)} = \cdots = x^{(l)}$.

Soit h l'indice le plus grand quittant J à une des itérations $k, \ldots, l-1$, par exemple à l'itération p. L'indice h a dû aussi être ajouté à J à une itération $q \in \{k, \ldots, l-1\}$. Soit y' le vecteur y à l'itération p, et soit w' le vecteur w à l'itération q. $y'Aw' = cw' > 0$. Soit alors r un indice pour lequel $y'_r a_r w' > 0$. Puisque $y'_r \neq 0$, l'indice r appartient à $J^{(p)}$. Si $r > h$, l'indice r doit aussi appartenir à $J^{(q)}$ et $J^{(q+1)}$, ce qui implique $a_r w' = 0$. Donc $r \leq h$. Mais par le choix de i à l'itération p, $y'_r < 0$ si et seulement si $r = h$, et par le choix de j à l'itération q, $a_r w' > 0$ si et seulement si $r = h$ (rappelons que $\lambda = 0$ et $a_r x^{(q)} = a_r x^{(p)} = \beta_r$ puisque $r \in J^{(p)}$). Cela conduit à une contradiction. □

Klee et Minty [1972] ainsi qu'Avis et Chvátal [1978] ont donné des exemples à n variables et $2n$ contraintes pour lesquels l'ALGORITHME DU SIMPLEXE (avec la règle de Bland) nécessite 2^n itérations, montrant ainsi la non-polynomialité de cet algorithme. On ne sait pas s'il existe une règle du pivot qui conduit à un algorithme polynomial. Cependant, Borgwardt [1982] a montré que le temps de calcul moyen (pour des instances aléatoires construites à partir d'un modèle probabiliste) peut être borné par un polynôme. Spielman et Teng [2004] ont introduit un modèle

d'analyse lisse : pour chaque input ils étudient l'espérance du temps de calcul quand on perturbe localement et de manière aléatoire cet input. Le maximum de toutes ces espérances est polynomialement borné. Kelner et Spielman [2006] ont proposé un algorithme randomisé polynomial pour la PROGRAMMATION LINÉAIRE qui ressemble à l'ALGORITHME DU SIMPLEXE. L'ALGORITHME DU SIMPLEXE est très rapide en pratique quand il est bien programmé ; voir paragraphe 3.3.

Nous allons montrer maintenant comment résoudre un PL quelconque avec l'ALGORITHME DU SIMPLEXE. Plus précisément, il nous faut montrer comment trouver un sommet initial. Puisqu'il existe des polyèdres sans aucun sommet, nous mettrons d'abord un PL donné sous une forme différente.

Soit $\max\{cx : Ax \leq b\}$ un PL. Nous remplaçons x par $y - z$ et nous utilisons la forme équivalente

$$\max \left\{ \begin{pmatrix} c & -c \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} : \begin{pmatrix} A & -A \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} \leq b,\ y, z \geq 0 \right\}.$$

On peut donc supposer que notre PL a la forme

$$\max\{cx : A'x \leq b',\ A''x \leq b'',\ x \geq 0\} \tag{3.3}$$

avec $b' \geq 0$ et $b'' < 0$. Nous appliquons d'abord l'ALGORITHME DU SIMPLEXE sur l'instance

$$\min\{(\mathbb{1}A'')x + \mathbb{1}y : A'x \leq b',\ A''x + y \geq b'',\ x, y \geq 0\}, \tag{3.4}$$

où $\mathbb{1}$ désigne un vecteur dont toutes les composantes sont égales à 1. Puisque $\binom{x}{y} = 0$ est un sommet, cela est possible. Ce PL n'est pas non borné puisque le minimum vaut au moins $\mathbb{1}b''$. Pour toute solution réalisable x de (3.3), $\binom{x}{b''-A''x}$ est une solution optimale de (3.4) ayant pour valeur $\mathbb{1}b''$. Donc, si le minimum de (3.4) est supérieur ou égal à $\mathbb{1}b''$, (3.3) est non réalisable.

Dans le cas contraire, soit $\binom{x}{y}$ un sommet optimal de (3.4) ayant pour valeur $\mathbb{1}b''$. Montrons que x est un sommet du polyèdre défini dans (3.3) : observons d'abord que $A''x + y = b''$. Soit n la dimension de x et m celle de y ; par la proposition 3.9, il existe un ensemble S de $n + m$ inégalités (3.4) satisfaites avec égalité, tel que la sous-matrice induite par ces $n + m$ inégalités soit non singulière.

Soit S' l'ensemble des inégalités de $A'x \leq b'$ et de $x \geq 0$ qui appartiennent à S. Soit S'' l'ensemble des inégalités de $A''x \leq b''$ tel que les inégalités correspondantes de $A''x + y \geq b''$ et de $y \geq 0$ appartiennent à S. $|S' \cup S''| \geq |S| - m = n$, et les inégalités $S' \cup S''$ sont linéairement indépendantes et satisfaites par x avec égalité. Donc x satisfait à égalité n inégalités linéairement indépendantes (3.3) ; x est donc un sommet. On peut initialiser l'ALGORITHME DU SIMPLEXE avec (3.3) et x.

3.3 Implémentation de l'algorithme du simplexe

La description précédente de l'ALGORITHME DU SIMPLEXE est simple, mais n'est pas utilisable pour une implémentation effective. Comme nous le verrons, il

n'est pas nécessaire de résoudre un système linéaire à chaque itération. Commençons, à titre de motivation, par la proposition suivante qui ne sera d'ailleurs pas utile par la suite ; si un PL est sous la forme $\max\{cx : Ax = b, x \geq 0\}$, on peut représenter chaque sommet non seulement par un sous-ensemble de lignes, mais aussi par un sous-ensemble de colonnes.

Si A est une matrice et J est un ensemble d'indices de colonnes, on notera par A^J la sous-matrice de A induite par les colonnes indicées par les éléments de J et par A^J_I est la sous-matrice de A induite par les colonnes indicées par les éléments de J et les lignes indicées par I. Il se peut que l'ordre des lignes et des colonnes soit important : si $J = (j_1, \ldots, j_k)$ est un vecteur d'indices de lignes (colonnes), A_J (A^J) sera la matrice dont la i-ième ligne (colonne) est la j_i-ième ligne (colonne) de A ($i = 1, \ldots, k$).

Proposition 3.15. *Soit $P := \{x : Ax = b, x \geq 0\}$, où A est une matrice et b un vecteur. x est un sommet de P si et seulement si $x \in P$ et si les colonnes de A associées aux composantes positives de x sont linéairement indépendantes.*

Preuve. Soit A une matrice $m \times n$. Soient $X := \left(\begin{smallmatrix} -I & 0 \\ A & I \end{smallmatrix}\right)$ et $b' := \left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right)$. Soient $N := \{1, \ldots, n\}$ et $M := \{n+1, \ldots, n+m\}$. Si $J \subseteq N \cup M$ est un ensemble d'indices tel que $|J| = n$, posons $\bar{J} := (N \cup M) \setminus J$. Alors X^N_J est non singulière si et seulement si $X^{N \cap \bar{J}}_{M \cap J}$ est non singulière et si et seulement si $X^{\bar{J}}_M$ est non singulière.

Si x est un sommet de P, il existe – par la proposition 3.9 – un ensemble $J \subseteq N \cup M$ tel que $|J| = n$, X^N_J soit non singulière et $X^N_J x = b'_J$. Alors les composantes de x correspondant à $N \cap J$ valent zéro. De plus, $X^{\bar{J}}_M$ est non singulière, et par conséquent les colonnes de $A^{N \cap \bar{J}}$ sont linéairement indépendantes.

Inversement, soit $x \in P$, et supposons que l'ensemble des colonnes de A associées aux composantes positives de x soient linéairement indépendantes. En ajoutant convenablement des vecteurs colonnes unité à ces colonnes, on obtient une sous-matrice non singulière X^B_M avec $x_i = 0$ pour $i \in N \setminus B$. X^N_B est non singulière et $X^N_B x = b'_B$. Donc, par la proposition 3.9, x est un sommet de P. □

Corollaire 3.16. *Soit $\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) \in P := \{\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) : Ax + y = b,\ x \geq 0,\ y \geq 0\}$. Alors $\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$ est un sommet de P si et seulement si les colonnes de $(A\, I)$ associées aux composantes positives de $\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$ sont linéairement indépendantes. De plus, x est un sommet de $\{x : Ax \leq b,\ x \geq 0\}$ si et seulement si $\left(\begin{smallmatrix} x \\ b - Ax \end{smallmatrix}\right)$ est un sommet de P.* □

Nous allons maintenant analyser le comportement de l'ALGORITHME DU SIMPLEXE quand on l'applique à un PL de la forme $\max\{cx : Ax \leq b,\ x \geq 0\}$.

Théorème 3.17. *Soient $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ et $c \in \mathbb{R}^n$. Soient $A' := \left(\begin{smallmatrix} -I \\ A \end{smallmatrix}\right)$, $b' := \left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right)$ et $\bar{c} := (c^\top, 0)$. Soit $B \in \{1, \ldots, n+m\}^m$ tel que $(A\, I)^B$ soit non singulière. Soit $J \subseteq \{1, \ldots, n+m\}$ l'ensemble des n indices restants. Soit $Q_B := ((A\, I)^B)^{-1}$.*

Alors :

(a) *A'_J est non singulière.*

(b) $(b' - A'x)_J = 0$, $(b' - A'x)_B = Q_B b$ *et* $c^\top x = \bar{c}^B Q_B b$, *où* $x := (A'_J)^{-1} b'_J$.

(c) *Soit* y *le vecteur tel que* $y_B = 0$ *et* $y^\top A' = c^\top$. *Alors* $y^\top = \bar{c}^B Q_B (A\, I) - \bar{c}$.

(d) *Soit* $i \in J$. *Soit* w *le vecteur tel que* $A'_i w = -1$ *et* $A'_{J\setminus\{i\}} w = 0$ *; on a alors* $A'_B w = Q_B (A\, I)^i$.

(e) *Considérons le tableau suivant :*

$$T_B := \left(\begin{array}{c|c} Q_B(A\, I) & Q_B b \\ \hline \bar{c}^B Q_B (A\, I) - \bar{c} & c^\top x \end{array} \right).$$

*B et T_B étant donnés, on peut calculer B' et $T_{B'}$ en un temps $O(m(n+m))$ où B' s'obtient à partir de B en remplaçant j par i, i et j étant donnés comme dans les étapes ②-④ de l'*ALGORITHME DU SIMPLEXE *appliqué à A', b', c, et à l'ensemble d'indices J.*

T_B est appelé le **tableau du simplexe** associé à la **base** B.

Preuve. (a) : soit $N := \{1, \ldots, n\}$. Comme $(A\, I)^B$ est non singulière, $(A')^{N\setminus J}_{J\setminus N}$ est aussi non singulière, et donc A'_J est non singulière.

(b) : la première affirmation découle directement de $A'_J x = b'_J$. D'autre part, $b = Ax + I(b - Ax) = (A\, I)(b' - A'x) = (A\, I)^B (b' - A'x)_B$ et $c^\top x = \bar{c}(b' - A'x) = \bar{c}^B (b' - A'x)_B = \bar{c}^B Q_B b$.

(c) : cela découle de $(\bar{c}^B Q_B (A\, I) - \bar{c})^B = \bar{c}^B Q_B (A\, I)^B - \bar{c}^B = 0$ et $(\bar{c}^B Q_B (A\, I) - \bar{c}) A' = \bar{c}^B Q_B (A\, I) A' - c^\top (-I) = c^\top$.

(d) : cela découle de $0 = (A\, I) A' w = (A\, I)^B (A'_B w) + (A\, I)^{J\setminus\{i\}} (A'_{J\setminus\{i\}} w) + (A\, I)^i (A'_i w) = (A\, I)^B (A'_B w) - (A\, I)^i$.

(e) : par (c), y est donné comme en ② de l'ALGORITHME DU SIMPLEXE par la dernière ligne de T_B. Si $y \geq 0$, l'algorithme s'arrête (x et y sont solutions optimales). Sinon i est le premier indice tel que $y_i < 0$, trouvé en un temps $O(n+m)$. Si la i-ième colonne de T_B n'a aucune composante positive, nous arrêtons (le PL est non borné, et w est donné par (d)). Sinon, par (b) et (d), λ dans ④ de l'ALGORITHME DU SIMPLEXE est égal à

$$\lambda = \min \left\{ \frac{(Q_B b)_j}{(Q_B (A\, I)^i)_j} : j \in \{1, \ldots, m\},\ (Q_B (A\, I)^i)_j > 0 \right\},$$

et j est, parmi les indices atteignant le minimum, celui pour lequel la j-ième composante de B est minimum. On peut donc calculer j en un temps $O(m)$ en considérant la i-ième et la dernière colonne de T_B. Cela fournit B'.

Nous pouvons calculer le tableau réactualisé $T_{B'}$ de la manière suivante : diviser la ligne j-ième par le coefficient de la ligne j et de la colonne i. Puis ajouter un multiple convenable de la j-ième ligne à toutes les autres lignes, de telle sorte que les composantes de la i-ième colonne soient égales à zéro sauf celle de la ligne j.

Notons que ces opérations sur les lignes ne détruisent pas la forme du tableau :

$$\left(\begin{array}{c|c} Q(A\, I) & Qb \\ \hline v(A\, I) - \bar{c} & vb \end{array} \right)$$

où Q est une matrice non singulière, v un vecteur, et où, en plus, $Q(A\,I)^{B'} = I$ et $(v(A\,I) - \bar{c})^{B'} = 0$. Puisqu'il n'y a qu'un choix possible pour Q et pour v : $Q = Q_{B'}$ et $v = \bar{c}^{B'}Q_{B'}$, la réactualisation du tableau $T_{B'}$ avec les opérations précédentes nécessite un temps $O(m(n+m))$. □

Pour commencer l'algorithme, nous considérons un PL sous la forme

$$\max\{cx : A'x \leq b',\ A''x \leq b'',\ x \geq 0\}$$

avec $A' \in \mathbb{R}^{m' \times n}$, $A'' \in \mathbb{R}^{m'' \times n}$, $b' \geq 0$ et $b'' < 0$. Nous exécutons d'abord l'ALGORITHME DU SIMPLEXE sur l'instance

$$\min\{(\mathbb{1}A'')x + \mathbb{1}y : A'x \leq b',\ A''x + y \geq b'',\ x, y \geq 0\},$$

en partant du tableau

$$\left(\begin{array}{cccc|c} A' & 0 & I & 0 & b' \\ -A'' & -I & 0 & I & -b'' \\ \hline \mathbb{1}A'' & \mathbb{1} & 0 & 0 & 0 \end{array}\right), \tag{3.5}$$

associé à la solution de base $x = 0$, $y = 0$. puis nous exécutons les itérations de l'ALGORITHME DU SIMPLEXE comme dans le théorème 3.17(e).

Si l'algorithme se termine avec la valeur optimale $\mathbb{1}b$, nous modifions ainsi le dernier tableau du simplexe : Multiplier certaines lignes par -1 de telle sorte qu'aucune des colonnes $n + m'' + m' + 1, \ldots, n + m'' + m' + m''$ (la quatrième partie de (3.5)) ne soit un vecteur unité, puis effacer la quatrième partie du tableau (c.-à-d. les colonnes $n + m'' + m' + 1, \ldots, n + m'' + m' + m''$), et remplacer la dernière ligne par $(-c, 0, 0, 0)$; enfin ajouter à la dernière ligne des multiples convenables des autres lignes de sorte à faire apparaître des zéros aux $m' + m''$ emplacements correspondant aux vecteurs colonnes unité ; cet ensemble d'indices sera notre base. Le résultat est le tableau du simplexe du PL original, associé à cette base. Nous pouvons donc poursuivre l'exécution de l'ALGORITHME DU SIMPLEXE comme dans le théorème 3.17(e).

Cela peut, d'ailleurs, s'effectuer de manière plus rapide. Supposons que l'on veuille résoudre un PL $\min\{cx : Ax \geq b,\ x \geq 0\}$ avec un très grand nombre d'inégalités qui sont implicitement données de telle sorte que l'on puisse résoudre efficacement le problème suivant : Un vecteur $x \geq 0$ étant donné, décider si $Ax \geq b$ ou alors trouver une inégalité violée. Nous appliquons l'ALGORITHME DU SIMPLEXE au PL dual $\max\{yb : yA \leq c, y \geq 0\} = \max\{by : A^\top y \leq c, y \geq 0\}$. Soit $\bar{b} := (b^\top, 0)$. Si B est une base, posons $Q_B := ((A^\top\, I)^B)^{-1}$ et nous ne gardons que la partie droite du tableau du simplexe

$$\left(\begin{array}{c|c} Q_B & Q_B c \\ \hline \bar{b}^B Q_B & b^\top x \end{array}\right).$$

La dernière ligne du tableau du simplexe complet est $\bar{b}^B Q_B(A^\top\, I) - \bar{b}$. Pour exécuter une itération, nous devons vérifier si $\bar{b}^B Q_B \geq 0$ et $\bar{b}^B Q_B A^\top - b \geq 0$,

puis trouver une composante négative quand une telle composante existe. Cela revient à résoudre le problème précédent pour $x = (\bar{b}^B Q_B)^\top$. Puis nous générons les colonnes du tableau du simplexe complet, mais seulement pour l'itération courante. Après avoir mis à jour le tableau réduit, nous pourrons de nouveau effacer le reste du tableau. Cette technique est connue sous le nom de *simplexe révisé* et de *génération de colonnes*. Nous étudierons ces applications ultérieurement.

3.4 Dualité

Le théorème 3.14 montre que les PL (3.1) (3.2) sont reliés entre eux. Cela justifie la définition suivante :

Définition 3.18. *Étant donné un PL* $\max\{cx : Ax \leq b\}$*, le* **PL dual** *est le PL* $\min\{yb : yA = c,\ y \geq 0\}$.

Le PL original $\max\{cx : Ax \leq b\}$ est alors souvent appelé le **PL primal**.

Proposition 3.19. *Le dual du dual d'un PL est équivalent au PL original.*

Preuve. Si le primal est $\max\{cx : Ax \leq b\}$, son dual est $\min\{yb : yA = c,\ y \geq 0\}$, ou de manière équivalente

$$-\max\left\{-by : \begin{pmatrix} A^\top \\ -A^\top \\ -I \end{pmatrix} y \leq \begin{pmatrix} c \\ -c \\ 0 \end{pmatrix}\right\}.$$

(Chaque contrainte d'égalité a été remplacée par deux inégalités.) Donc le dual du dual est

$$-\min\left\{zc - z'c : \begin{pmatrix} A & -A & -I \end{pmatrix} \begin{pmatrix} z \\ z' \\ w \end{pmatrix} = -b,\ z, z', w \geq 0\right\},$$

ce qui est équivalent à $-\min\{-cx : -Ax - w = -b,\ w \geq 0\}$ (nous avons remplacé x par $z' - z$). En éliminant les variables d'écart w nous obtenons une forme équivalente au primal initial. □

Nous allons énoncer maintenant le théorème le plus important de la PROGRAMMATION LINÉAIRE, le théorème de la dualité :

Théorème 3.20. (von Neumann [1947], Gale, Kuhn et Tucker [1951]) *Si les polyèdres* $P := \{x : Ax \leq b\}$ *et* $D := \{y : yA = c,\ y \geq 0\}$ *sont non vides, alors* $\max\{cx : x \in P\} = \min\{yb : y \in D\}$.

Preuve. Si D est non vide, il a un sommet y. Exécutons l'ALGORITHME DU SIMPLEXE sur $\min\{yb : y \in D\}$ et y. Par la proposition 3.13, $\min\{yb : y \in D\}$ n'est pas non borné puisque P admet une solution réalisable x. Donc par le théorème 3.14, l'ALGORITHME DU SIMPLEXE retournera une solution optimale y du PL $\min\{yb : y \in D\}$ notée y et une solution de son dual notée z. Cependant le dual est $\max\{cx : x \in P\}$ et, par la proposition 3.19, $yb = cz$. □

Il existe d'autres relations entre les solutions optimales du primal et du dual :

Corollaire 3.21. *Soit x une solution réalisable du PL $\max\{cx : Ax \leq b\}$ et soit y une solution réalisable de son dual $\min\{yb : yA = c,\ y \geq 0\}$ ($Ax \leq b$, $yA = c$ et $y \geq 0$). Alors les conditions suivantes sont équivalentes :*

(a) *x et y sont solutions optimales.*

(b) *$cx = yb$.*

(c) *$y(b - Ax) = 0$.*

Preuve. Le théorème de la dualité 3.20 implique l'équivalence de (a) et (b). L'équivalence de (b) et (c) se déduit de $y(b - Ax) = yb - yAx = yb - cx$. □

La condition (c) est souvent appelée **condition des écarts complémentaires**. Elle peut aussi se formuler de la manière suivante : un point $x^* \in P = \{x : Ax \leq b\}$ est une solution optimale de $\max\{cx : x \in P\}$ si et seulement si c est une combinaison non négative des seules lignes de A qui correspondent aux inégalités de $Ax \leq b$ satisfaites avec égalité par x^*. Cela implique également :

Corollaire 3.22. *Soit $P = \{x : Ax \leq b\}$ un polyèdre et $Z \subseteq P$. L'ensemble des vecteurs c pour lesquels $z \in Z$ est solution optimale de $\max\{cx : x \in P\}$ est le cône engendré par les lignes de A', où $A'x \leq b'$ est le sous-système maximal de $Ax \leq b$ tel que $A'z = b'$ pour tout $z \in Z$.*

Preuve. Il existe $z \in Z$ qui satisfait strictement toutes les autres inégalités de $Ax \leq b$. Soit c un vecteur pour lequel z est solution optimale de $\max\{cx : x \in P\}$. Par le corollaire 3.21 il existe y tel que $y \geq 0$ et $c = yA'$, c.-à-d. c est une combinaison linéaire non négative des lignes de A'.

Inversement, si $a'x \leq \beta'$ est une ligne de $A'x \leq b'$ et $z \in Z$, alors $a'z = \beta' = \max\{a'x : x \in P\}$. □

Écrivons le corollaire 3.21 sous une autre forme :

Corollaire 3.23. *Soient $\max\{cx : Ax \geq b,\ x \geq 0\}$ et $\min\{yb : yA \leq c,\ y \geq 0\}$ deux PL duaux. Soit x une solution réalisable du premier et y une solution réalisable du second : $Ax \leq b$, $yA = c$ et $x, y \geq 0$. Alors les conditions suivantes sont équivalentes :*

(a) *x et y sont solutions optimales.*

(b) *$cx = yb$.*

(c) *$(c - yA)x = 0$ et $y(b - Ax) = 0$.*

Preuve. L'équivalence de (a) et (b) s'obtient par l'application du théorème de la dualité 3.20 à $\max\left\{(-c)x : \left(\begin{smallmatrix}-A\\-I\end{smallmatrix}\right) x \leq \left(\begin{smallmatrix}-b\\0\end{smallmatrix}\right)\right\}$.

(b) et (c) sont équivalents, car $y(b-Ax) \leq 0 \leq (c-yA)x$ pour toute paire de solutions réalisables x, y et $y(b-Ax) = (c-yA)x$ si et seulement si $yb = cx$. □

Les deux conditions de (c) sont quelquefois appelées **condition des écarts complémentaires du primal** et du **dual**.

Le théorème de la dualité a de nombreuses applications en optimisation combinatoire. Son importance réside dans le fait que pour montrer qu'une solution réalisable d'un PL est optimale, il suffit d'exhiber une solution réalisable de son dual. Nous allons montrer maintenant comment prouver qu'un PL est non borné ou non réalisable :

Théorème 3.24. *Il existe un vecteur x tel que $Ax \leq b$ si et seulement si $yb \geq 0$ pour tout vecteur $y \geq 0$ tel que $yA = 0$.*

Preuve. S'il existe un vecteur x tel que $Ax \leq b$, alors $yb \geq yAx = 0$ pour tout $y \geq 0$ tel que $yA = 0$.

Soit le PL suivant :

$$-\min\{\mathbb{1}w : Ax - w \leq b,\ w \geq 0\}. \tag{3.6}$$

En l'écrivant sous forme standard il vient

$$\max\left\{\begin{pmatrix}0 & -\mathbb{1}\end{pmatrix}\begin{pmatrix}x\\w\end{pmatrix} : \begin{pmatrix}A & -I\\0 & -I\end{pmatrix}\begin{pmatrix}x\\w\end{pmatrix} \leq \begin{pmatrix}b\\0\end{pmatrix}\right\}.$$

Le dual de ce PL est

$$\min\left\{\begin{pmatrix}b & 0\end{pmatrix}\begin{pmatrix}y\\z\end{pmatrix} : \begin{pmatrix}A^\top & 0\\-I & -I\end{pmatrix}\begin{pmatrix}y\\z\end{pmatrix} = \begin{pmatrix}0\\-\mathbb{1}\end{pmatrix},\ y, z \geq 0\right\},$$

ou, de manière équivalente,

$$\min\{yb : yA = 0,\ 0 \leq y \leq \mathbb{1}\}. \tag{3.7}$$

Puisque (3.6) et (3.7) ont une solution ($x = 0$, $w = |b|$, $y = 0$), nous pouvons appliquer le théorème 3.20. Les valeurs optimales de (3.6) et (3.7) sont donc égales. Comme le système $Ax \leq b$ a une solution si et seulement si la valeur optimale de (3.6) est zéro, le résultat est démontré. □

On peut donc prouver qu'un système d'inégalités linéaires $Ax \leq b$ n'a pas de solutions en fournissant un vecteur $y \geq 0$ tel que $yA = 0$ et $yb < 0$. Mentionnons deux formulations équivalentes du théorème 3.24 :

Corollaire 3.25. *Il existe un vecteur $x \geq 0$ tel que $Ax \leq b$ si et seulement si $yb \geq 0$ pour tout vecteur $y \geq 0$ tel que $yA \geq 0$.*

Preuve. Appliquons le théorème 3.24 au système $\binom{A}{-I} x \leq \binom{b}{0}$. □

Corollaire 3.26. (Farkas [1894]) *Il existe un vecteur $x \geq 0$ tel que $Ax = b$ si et seulement si $yb \geq 0$ pour tout vecteur y tel que $yA \geq 0$.*

Preuve. Appliquons le corollaire 3.25 au système $\binom{A}{-A} x \leq \binom{b}{-b}, x \geq 0$. □

Le corollaire 3.26 est connu sous le nom de lemme de Farkas. Les résultats précédents impliquent en retour le théorème de la dualité 3.20, ce qui est intéressant, car on peut les démontrer directement de manière simple (ils étaient connus en fait avant l'ALGORITHME DU SIMPLEXE) ; voir les exercices 10 et 11.

Nous avons vu comment prouver qu'un PL est non réalisable. Comment prouver qu'un PL est non borné ? Le théorème suivant répond à cette question.

Théorème 3.27. *Si un PL est non borné, alors son dual est non réalisable. Si un PL admet une solution optimale, son dual admet aussi une solution optimale.*

Preuve. La première affirmation se déduit de la proposition 3.13.

Pour montrer la seconde, supposons que le PL (primal) $\max\{cx : Ax \leq b\}$ ait une solution optimale x^*, mais que le dual $\min\{yb : yA = c, y \geq 0\}$ soit non réalisable (d'après ce qui précède, il ne peut être non borné). Il n'existe donc aucun $y \geq 0$ tel que $A^\top y = c$, et nous pouvons appliquer le lemme de Farkas (corollaire 3.26) pour obtenir un vecteur z tel que $zA^\top \geq 0$ et $zc < 0$. Mais alors $x^* - z$ est réalisable pour le primal, car $A(x^* - z) = Ax^* - Az \leq b$. En observant que $c(x^* - z) > cx^*$ nous contredisons l'optimalité de x^*. □

Il y a donc quatre possibilités pour une paire primale-duale de PLs : soit les deux ont une solution optimale (de même valeur), soit l'un d'entre eux est non réalisable tandis que l'autre est non borné, soient les deux sont non réalisables. Notons également le résultat suivant :

Corollaire 3.28. *Un PL $\max\{cx : Ax \leq b\}$ est borné si et seulement si c appartient au cône engendré par les lignes de A.*

Preuve. Le PL est borné si et seulement si son dual est réalisable, c.-à-d. s'il existe $y \geq 0$ tel que $yA = c$. □

Le lemme de Farkas nous permet de montrer que tout cône finiment engendré est polyédral :

Théorème 3.29. (Minkowski [1896], Weyl [1935]) *Un cône est polyédral si et seulement s'il est finiment engendré.*

Preuve. La condition suffisante est obtenue par le lemme 3.12. Considérons alors le cône C engendré par $a_1, \ldots, a_t$ et montrons qu'il est polyédral. Soit A la matrice dont les lignes sont $a_1, \ldots, a_t$.

Par le lemme 3.12, le cône $D := \{x : Ax \leq 0\}$ est engendré par des vecteurs $b_1, \ldots, b_s$. Soit B la matrice dont les lignes sont $b_1, \ldots, b_s$. Montrons que $C = \{x : Bx \leq 0\}$.

Comme $b_j a_i = a_i b_j \leq 0$ pour tout i et j, nous avons $C \subseteq \{x : Bx \leq 0\}$. Supposons qu'il existe un vecteur $w \notin C$ tel que $Bw \leq 0$. $w \notin C$ signifie qu'il n'existe aucun $v \geq 0$ tel que $A^\top v = w$. Par le lemme de Farkas (corollaire 3.26) cela signifie qu'il existe un vecteur y tel que $yw < 0$ et $Ay \geq 0$. Donc $-y \in D$. Puisque D est engendré par $b_1, \ldots, b_s$ $-y = zB$ pour un vecteur $z \geq 0$. Mais alors $0 < -yw = zBw \leq 0$, ce qui est une contradiction. □

3.5 Enveloppes convexes et polytopes

Dans ce paragraphe, nous présentons quelques résultats reliés aux polytopes. En particulier, nous montrons que les polytopes sont les ensembles qui sont l'enveloppe convexe d'un nombre fini de points. Rappelons d'abord quelques définitions :

Définition 3.30. *Soient k vecteurs $x_1, \ldots, x_k \in \mathbb{R}^n$ et $\lambda_1, \ldots, \lambda_k \geq 0$ tels que $\sum_{i=1}^k \lambda_i = 1$, nous dirons que $x = \sum_{i=1}^k \lambda_i x_i$ est une* **combinaison convexe** *de $x_1, \ldots, x_k$. $X \subseteq \mathbb{R}^n$ est* **convexe** *si $\lambda x + (1-\lambda)y \in X$ pour $x, y \in X$ et $\lambda \in [0,1]$. L'***enveloppe convexe** $\mathrm{conv}(X)$ *de X est l'ensemble des combinaisons convexes des points de X. $x \in X$ est un* **point extrême** *de X si $x \notin \mathrm{conv}(X \setminus \{x\})$.*

X est convexe si et seulement si toutes les combinaisons convexes des points de X sont aussi dans X. L'enveloppe convexe de X est le plus petit ensemble convexe contenant X. L'intersection d'ensembles convexes est convexe. Les polyèdres sont donc convexes. Montrons maintenant le «théorème de la base finie pour les polytopes», résultat fondamental dont la preuve directe est assez élaborée :

Théorème 3.31. (Minkowski [1896], Steinitz [1916], Weyl [1935]) *P est un polytope si et seulement s'il est l'enveloppe convexe d'un nombre fini de points.*

Preuve. (Schrijver [1986]) Soit $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ un polytope non vide. Il est évident que

$$P = \left\{ x : \begin{pmatrix} x \\ 1 \end{pmatrix} \in C \right\}, \text{ où } C = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \in \mathbb{R}^{n+1} : \lambda \geq 0,\ Ax - \lambda b \leq 0 \right\}.$$

C est un cône polyédral, et par le théorème 3.29, C est engendré par un nombre fini de vecteurs non nuls, $\binom{x_1}{\lambda_1}, \ldots, \binom{x_k}{\lambda_k}$. Puisque P est borné, aucun des λ_i n'est nul et on peut tous les supposer égaux à 1. Donc $x \in P$ si et seulement si

$$\begin{pmatrix} x \\ 1 \end{pmatrix} = \mu_1 \begin{pmatrix} x_1 \\ 1 \end{pmatrix} + \cdots + \mu_k \begin{pmatrix} x_k \\ 1 \end{pmatrix}$$

avec $\mu_1, \ldots, \mu_k \geq 0$. Autrement dit, P est l'enveloppe convexe de $x_1, \ldots, x_k$.

Supposons maintenant que P soit l'enveloppe convexe de $x_1, \ldots, x_k \in \mathbb{R}^n$. Alors $x \in P$ si et seulement si $\binom{x}{1} \in C$, où C est le cône engendré par $\binom{x_1}{1}, \ldots, \binom{x_k}{1}$. Par le théorème 3.29, C est polyédral et

$$C = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} : Ax + b\lambda \leq 0 \right\}.$$

On a donc $P = \{x \in \mathbb{R}^n : Ax + b \leq 0\}$. □

Corollaire 3.32. *Un polytope est l'enveloppe convexe de ses sommets.*

Preuve. Soit P un polytope. Par le théorème 3.31, l'enveloppe convexe de ses sommets est un polytope Q et $Q \subseteq P$. Supposons qu'il existe $z \in P \setminus Q$. Alors il existe un vecteur c tel que $cz > \max\{cx : x \in Q\}$. L'hyperplan support $\{x : cx = \max\{cy : y \in P\}\}$ de P induit une face de P ne contenant aucun sommet. Cela est impossible par le corollaire 3.10. □

Les deux résultats précédents ainsi que le suivant sont le point de départ de la combinatoire polyédrale ; ils seront souvent utilisés dans ce livre. Pour un ensemble de base E et un sous-ensemble $X \subseteq E$, le **vecteur d'incidence** de X (par rapport à E) est le vecteur $x \in \{0,1\}^E$ avec $x_e = 1$ pour $e \in X$ et $x_e = 0$ pour $e \in E \setminus X$.

Corollaire 3.33. *Soit $(E, \mathcal{F})$ un système d'ensembles, soit P l'enveloppe convexe des vecteurs d'incidence des éléments de $\mathcal{F}$, et soit $c : E \to \mathbb{R}$. Alors* $\max\{cx : x \in P\} = \max\{c(X) : X \in \mathcal{F}\}$.

Preuve. Puisque $\max\{cx : x \in P\} \geq \max\{c(X) : X \in \mathcal{F}\}$, soit x une solution optimale de $\max\{cx : x \in P\}$ (P est un polytope par le théorème 3.31). Par définition de P, x est combinaison convexe des vecteurs d'incidence $y_1, \ldots, y_k$ des éléments de $\mathcal{F}$: $x = \sum_{i=1}^k \lambda_i y_i$ avec $\lambda_1, \ldots, \lambda_k \geq 0$ et $\sum_{i=1}^k \lambda_i = 1$. Puisque $cx = \sum_{i=1}^k \lambda_i cy_i$, $cy_i \geq cx$ pour au moins un $i \in \{1, \ldots, k\}$. Cet y_i est le vecteur d'incidence d'un ensemble $Y \in \mathcal{F}$ tel que $c(Y) = cy_i \geq cx$. □

Exercices

1. Soient H un hypergraphe, $F \subseteq V(H)$, et $x, y : F \to \mathbb{R}$. Il s'agit de trouver $x, y : V(H) \setminus F \to \mathbb{R}$ tel que $\sum_{e \in E(H)} (\max_{v \in e} x(v) - \min_{v \in e} x(v) + \max_{v \in e} y(v) - \min_{v \in e} y(v))$ soit minimum. Montrer que ce problème peut se formuler comme un PL.
   Note : cela est une relaxation d'un problème de placement dans le domaine de la conception des circuits intégrés. H est ici la «netlist», et ses sommets correspondent aux modules à placer sur la puce. Certains (ceux de F) sont prépositionnés. La principale difficulté, ignorée dans cette relaxation, est que les modules ne doivent pas se chevaucher.
2. Nous dirons que des vecteurs $x_1, \ldots, x_k$ sont affinement indépendants s'il n'existe aucun $\lambda \in \mathbb{R}^k \setminus \{0\}$ avec $\lambda^\top \mathbb{1} = 0$ tel que $\sum_{i=1}^k \lambda_i x_i = 0$. Soit $\emptyset \neq X \subseteq \mathbb{R}^n$. Montrer que la cardinalité maximum d'un ensemble de vecteurs affinement indépendants de X est égale à $\dim X + 1$.

3. Soient $P, Q \in \mathbb{R}^n$ deux polyèdres. Montrer que la fermeture de $\mathrm{conv}(P \cup Q)$ est un polyèdre. Donner un exemple où $\mathrm{conv}(P \cup Q)$ n'est pas un polyèdre.
4. Montrer que la recherche de la plus grande boule incluse dans un polyèdre peut se formuler comme un PL.
5. Soit P un polyèdre. Montrer que la dimension de toute facette de P est égale à $\dim P - 1$.
6. Soit F une face minimale d'un polyèdre $\{x : Ax \leq b\}$. Montrer que, pour toute paire $x, y \in F$, $Ax = Ay$.
7. Écrire le dual du PL (1.1) formulant le PROBLÈME D'AFFECTATION DES TÂCHES. Montrer comment résoudre simplement le primal et le dual quand il y a deux tâches.
8. Soit G un graphe orienté, $c : E(G) \to \mathbb{R}_+$, $E_1, E_2 \subseteq E(G)$, et $s, t \in V(G)$. Soit le PL suivant :

$$\begin{array}{lrcll} \min & \displaystyle\sum_{e \in E(G)} c(e) y_e & & & \\ \text{s.c.} & y_e & \geq & z_w - z_v & (e = (v,w) \in E(G)) \\ & z_t - z_s & = & 1 & \\ & y_e & \geq & 0 & (e \in E_1) \\ & y_e & \leq & 0 & (e \in E_2). \end{array}$$

   Montrer qu'il existe une solution optimale (y, z) et $s \in X \subseteq V(G) \setminus \{t\}$ vérifiant $y_e = 1$ si $e \in \delta^+(X)$, $y_e = -1$ si $e \in \delta^-(X) \setminus E_1$, et $y_e = 0$ pour toutes les autres arêtes e.
   Indication : utiliser les conditions des écarts complémentaires pour les arêtes entrant dans ou sortant de $\{v \in V(G) : z_v \leq z_s\}$.
9. Soit $Ax \leq b$ un système d'inégalités linéaires de n variables. En multipliant chaque ligne par une constante positive, on peut supposer que les composantes de la première colonne de A valent 0, -1 et 1. $Ax \leq b$ s'écrit donc :

$$\begin{array}{rcll} a_i' x' & \leq & b_i & (i = 1, \ldots, m_1), \\ -x_1 + a_j' x' & \leq & b_j & (j = m_1 + 1, \ldots, m_2), \\ x_1 + a_k' x' & \leq & b_k & (k = m_2 + 1, \ldots, m), \end{array}$$

   où $x' = (x_2, \ldots, x_n)$ et $a_1', \ldots, a_m'$ sont les lignes de A sans le premier coefficient. On peut alors éliminer x_1 : montrer que $Ax \leq b$ a une solution si et seulement si le système

$$\begin{array}{rcll} a_i' x' & \leq & b_i & (i = 1, \ldots, m_1), \\ a_j' x' - b_j & \leq & b_k - a_k' x' & (j = m_1 + 1, \ldots, m_2,\ k = m_2 + 1, \ldots, m) \end{array}$$

a une solution. Montrer qu'en réitérant cette opération, on obtient un algorithme qui résout le système d'inégalités $Ax \leq b$ ou qui prouve sa non-réalisabilité.
Note : cette méthode est connue sous le nom de méthode d'élimination de Fourier-Motzkin, car elle a été proposée par Fourier et étudiée par Motzkin [1936]. On peut montrer que cet algorithme n'est pas polynomial.

10. Utiliser l'élimination de Fourier-Motzkin (exercice 9) pour prouver le théorème 3.24 directement.
(Kuhn [1956])

11. Montrer que le théorème 3.24 implique le théorème de la dualité 3.20.

12. Montrer le théorème de décomposition des polyèdres : tout polyèdre P peut s'écrire $P = \{x + c : x \in X,\ c \in C\}$, où X est un polytope et C est un cône polyédral.
(Motzkin [1936])

∗ 13. Soit P un polyèdre rationnel et F une face de P. Montrer que

$$\{c : cz = \max\{cx : x \in P\} \text{ pour tout } z \in F\}$$

est un cône polyédral rationnel.

14. Montrer le théorème de Carathéodory :
si $X \subseteq \mathbb{R}^n$ et $y \in \text{conv}(X)$, il existe $x_1, \ldots, x_{n+1} \in X$ tels que $y \in \text{conv}(\{x_1, \ldots, x_{n+1}\})$.
(Carathéodory [1911])

15. Montrer l'extension suivante du théorème de Carathéodory (exercice 14) :
si $X \subseteq \mathbb{R}^n$ et $y, z \in \text{conv}(X)$, il existe $x_1, \ldots, x_n \in X$ tel que $y \in \text{conv}(\{z, x_1, \ldots, x_n\})$.

16. Montrer que les points extrêmes d'un polyèdre sont ses sommets.

17. Soit P un polytope non vide. Considérons le graphe $G(P)$ dont les sommets sont les sommets de P et dont les arêtes sont associées aux 1-faces de P. Soit x un sommet de P, et c un vecteur tel que $c^\top x < \max\{c^\top z : z \in P\}$. Montrer qu'il existe un voisin y de x dans $G(P)$ tel que $c^\top x < c^\top y$.

∗ 18. Utiliser l'exercice 17 pour montrer que $G(P)$ est n-connexe pour tout polytope P de dimension ($n \geq 1$).

19. Soit $P \subseteq \mathbb{R}^n$ un polytope (non nécessairement rationnel) et soit $y \notin P$. Montrer qu'il existe un vecteur c tel que $\max\{cx : x \in P\} < cy$. Montrer que cette condition n'est pas, en général, vérifiée pour les polyèdres.

20. Soient $X \subset \mathbb{R}^n$ convexe non vide, $\bar{X}$ la fermeture de X, et $y \notin X$. Montrer :
    (a) Il existe un unique point de $\bar{X}$ qui est à distance minimum de y.
    (b) Il existe un vecteur $a \in \mathbb{R}^n \setminus \{0\}$ vérifiant $a^\top x \leq a^\top y$ pour tout $x \in X$.
    (c) Si X est borné et $y \notin \bar{X}$, alors il existe un vecteur $a \in \mathbb{Q}^n$ vérifiant $a^\top x < a^\top y$ pour tout $x \in X$.
    (d) Un ensemble convexe fermé est l'intersection de tous les demi-espaces fermés le contenant.

Références

Littérature générale :

Bertsimas, D., Tsitsiklis, J.N. [1997] : Introduction to Linear Optimization. Athena Scientific, Belmont 1997

Chvátal, V. [1983] : Linear Programming. Freeman, New York 1983

Matoušek, J., Gärtner, B. [2007] : Understanding and Using Linear Programming. Springer, Berlin 2007

Padberg, M. [1999] : Linear Optimization and Extensions. Second Edition. Springer, Berlin 1999

Schrijver, A. [1986] : Theory of Linear and Integer Programming. Wiley, Chichester 1986

Références citées :

Avis, D., Chvátal, V. [1978] : Notes on Bland's pivoting rule. Mathematical Programming Study 8 (1978), 24–34

Bland, R.G. [1977] : New finite pivoting rules for the simplex method. Mathematics of Operations Research 2 (1977), 103–107

Borgwardt, K.-H. [1982] : The average number of pivot steps required by the simplex method is polynomial. Zeitschrift für Operations Research 26 (1982), 157–177

Carathéodory, C. [1911] : Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. Rendiconto del Circolo Matematico di Palermo 32 (1911), 193–217

Dantzig, G.B. [1951] : Maximization of a linear function of variables subject to linear inequalities. In : Activity Analysis of Production and Allocation (T.C. Koopmans, ed.), Wiley, New York 1951, pp. 359–373

Dantzig, G.B., Orden, A., Wolfe, P. [1955] : The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific Journal of Mathematics 5 (1955), 183–195

Farkas, G. [1894] : A Fourier-féle mechanikai elv alkalmazásai. Mathematikai és Természettudományi Értesitö 12 (1894), 457–472

Gale, D., Kuhn, H.W., Tucker, A.W. [1951] : Linear programming and the theory of games. In : Activity Analysis of Production and Allocation (T.C. Koopmans, ed.), Wiley, New York 1951, pp. 317–329

Hoffman, A.J., Kruskal, J.B. [1956] : Integral boundary points of convex polyhedra. In : Linear Inequalities and Related Systems ; Annals of Mathematical Study 38 (H.W. Kuhn, A.W. Tucker, eds.), Princeton University Press, Princeton 1956, pp. 223–246

Kelner, J.A., Spielman, D.A. [2006] : A randomized polynomial-time simplex algorithm for linear programming. Proceedings of the 38th Annual ACM Symposium on Theory of Computing (2006), 51–60

Klee, V., Minty, G.J. [1972] : How good is the simplex algorithm ? In : Inequalities III (O. Shisha, ed.), Academic Press, New York 1972, pp. 159–175

Kuhn, H.W. [1956] : Solvability and consistency for linear equations and inequalities. The American Mathematical Monthly 63 (1956), 217–232

Minkowski, H. [1896] : Geometrie der Zahlen. Teubner, Leipzig 1896

Motzkin, T.S. [1936] : Beiträge zur Theorie der linearen Ungleichungen (Dissertation). Azriel, Jerusalem 1936

von Neumann, J. [1947] : Discussion of a maximum problem. Working paper. Published in : John von Neumann, Collected Works ; Vol. VI (A.H. Taub, ed.), Pergamon Press, Oxford 1963, pp. 27–28

Spielman, D.A., Teng, S.-H. [2004] : Smoothed analysis of algorithms : why the simplex algorithm usually takes polynomial time. Journal of the ACM 51 (2004), 385–463

Steinitz, E. [1916] : Bedingt konvergente Reihen und konvexe Systeme. Journal für die reine und angewandte Mathematik 146 (1916), 1–52

Weyl, H. [1935] : Elementare Theorie der konvexen Polyeder. Commentarii Mathematici Helvetici 7 (1935), 290–306

Chapitre 4

Algorithmes de programmation linéaire

Il y a essentiellement trois types d'algorithmes en PROGRAMMATION LINÉAIRE : l'ALGORITHME DU SIMPLEXE (voir paragraphe 3.2), les algorithmes de points intérieurs et la MÉTHODE DES ELLIPSOÏDES.

Chacune de ces méthodes a un inconvénient : on ne connaît aucune variante de l'ALGORITHME DU SIMPLEXE ayant un temps de calcul polynomial, contrairement aux deux autres méthodes. Nous présenterons la MÉTHODE DES ELLIPSOÏDES qui résout la PROGRAMMATION LINÉAIRE en temps polynomial aux paragraphes 4.4 et 4.5. Cependant, cette méthode n'a aucune efficacité pratique. Les algorithmes de points intérieurs et l'ALGORITHME DU SIMPLEXE malgré son temps de calcul exponentiel dans le pire des cas sont bien plus performants et sont tous deux utilisés en pratique. La MÉTHODE DES ELLIPSOÏDES et les algorithmes de points intérieurs peuvent être utilisés pour des problèmes généraux d'optimisation convexes, comme dans le cas de la programmation semi-définie.

Un avantage de l'ALGORITHME DU SIMPLEXE et de la MÉTHODE DES ELLIPSOÏDES est qu'ils n'exigent pas que le PL soit donné explicitement. Il suffit de disposer d'un oracle (un sous-programme) qui décide si un vecteur donné est réalisable et qui, dans le cas contraire, retourne une contrainte violée. Nous étudierons dans le détail cette propriété de la MÉTHODE DES ELLIPSOÏDES au paragraphe 4.6 car elle implique que de nombreux problèmes d'optimisation combinatoire peuvent se résoudre en temps polynomial ; c'est quelquefois la seule manière de le démontrer. C'est pour cela que nous décrirons la MÉTHODE DES ELLIPSOÏDES mais que nous n'étudierons pas les algorithmes de points intérieurs.

Quand un algorithme est polynomial, une solution optimale doit avoir une représentation binaire de longueur bornée par un polynôme par rapport à la taille de l'input. Nous montrerons au paragraphe 4.1 que cette propriété est vraie dans le cas de la PROGRAMMATION LINÉAIRE. Dans les paragraphes 4.2 et 4.3 nous rap-

pellerons quelques algorithmes de base dont nous aurons besoin ultérieurement, en particulier la méthode de Gauss pour résoudre les systèmes linéaires.

4.1 Taille des sommets et des faces

Les instances de PROGRAMMATION LINÉAIRE sont des vecteurs et des matrices. Puisque aucun algorithme fortement polynomial n'est connu pour la PROGRAMMATION LINÉAIRE, nous allons nous restreindre aux instances rationnelles dans l'analyse du temps de calcul. Nous supposerons que tous les nombres sont codés en binaire. Pour estimer la taille de cette représentation (le nombre de bits) nous écrirons $\text{taille}(n) := 1 + \lceil \log(|n| + 1) \rceil$ pour des entiers $n \in \mathbb{Z}$ et $\text{taille}(r) := \text{taille}(p) + \text{taille}(q)$ pour des nombres rationnels $r = \frac{p}{q}$, ou p, q sont des entiers relativement premiers (leur plus grand commun diviseur est 1). Pour des vecteurs $x = (x_1, \ldots, x_n) \in \mathbb{Q}^n$ nous stockons leurs composantes : $\text{taille}(x) := n + \text{taille}(x_1) + \ldots + \text{taille}(x_n)$. Pour une matrice $A \in \mathbb{Q}^{m \times n}$ avec coefficients a_{ij}, $\text{taille}(A) := mn + \sum_{i,j} \text{taille}(a_{ij})$.

Bien entendu ces valeurs précises peuvent paraître être un choix arbitraire, mais rappelons que nous ne prenons pas en compte les facteurs constants. Pour les algorithmes polynomiaux, il est important que les tailles des nombres obtenues après les opérations arithmétiques ne croissent pas trop vite. Notons que :

Proposition 4.1. *Si $r_1, \ldots, r_n$ sont des nombres rationnels, alors*

$$\begin{aligned} \text{taille}(r_1 \cdots r_n) &\leq \text{taille}(r_1) + \cdots + \text{taille}(r_n); \\ \text{taille}(r_1 + \cdots + r_n) &\leq 2(\text{taille}(r_1) + \cdots + \text{taille}(r_n)). \end{aligned}$$

Preuve. Pour des entiers $s_1, \ldots, s_n$, $\text{taille}(s_1 \cdots s_n) \leq \text{taille}(s_1) + \cdots + \text{taille}(s_n)$ et $\text{taille}(s_1 + \cdots + s_n) \leq \text{taille}(s_1) + \cdots + \text{taille}(s_n)$.

Soient $r_i = \frac{p_i}{q_i}$, où p_i et q_i sont des entiers non nuls ($i = 1, \ldots, n$). Alors $\text{taille}(r_1 \cdots r_n) \leq \text{taille}(p_1 \cdots p_n) + \text{taille}(q_1 \cdots q_n) \leq \text{taille}(r_1) + \cdots + \text{taille}(r_n)$.

Pour la seconde condition, observons que la taille du dénominateur $q_1 \cdots q_n$ est au plus $\text{taille}(q_1) + \cdots + \text{taille}(q_n)$. Le numérateur est la somme des nombres $q_1 \cdots q_{i-1} p_i q_{i+1} \cdots q_n$ ($i = 1, \ldots, n$) ; sa valeur absolue est au plus $(|p_1| + \cdots + |p_n|)|q_1 \cdots q_n|$. Donc la taille du numérateur est au plus $\text{taille}(r_1) + \cdots + \text{taille}(r_n)$. □

La première partie de cette proposition implique que nous pourrons supposer sans perte de généralité que les nombres des instances sont des entiers, car sinon on pourrait multiplier tous ces nombres par le produit des dénominateurs. Pour l'addition et le produit scalaire de vecteurs, nous avons les relations :

Proposition 4.2. *Si $x, y \in \mathbb{Q}^n$ sont des vecteurs rationnels, alors*

$$\begin{aligned} \text{taille}(x + y) &\leq 2(\text{taille}(x) + \text{taille}(y)); \\ \text{taille}(x^\top y) &\leq 2(\text{taille}(x) + \text{taille}(y)). \end{aligned}$$

Preuve. Par la proposition 4.1, $\text{taille}(x+y) = n + \sum_{i=1}^{n} \text{taille}(x_i + y_i) \leq n + 2\sum_{i=1}^{n} \text{taille}(x_i) + 2\sum_{i=1}^{n} \text{taille}(y_i) = 2(\text{taille}(x) + \text{taille}(y)) - 3n$ et $\text{taille}(x^\top y) = \text{taille}\left(\sum_{i=1}^{n} x_i y_i\right) \leq 2\sum_{i=1}^{n} \text{taille}(x_i y_i) \leq 2\sum_{i=1}^{n} \text{taille}(x_i) + 2\sum_{i=1}^{n} \text{taille}(y_i) = 2(\text{taille}(x) + \text{taille}(y)) - 4n$. □

Même avec des opérations plus compliquées, les nombres générés pendant l'algorithme ne croîtront pas trop vite. Rappelons que le déterminant d'une matrice $A = (a_{ij})_{1 \leq i,j \leq n}$ est :

$$\det A := \sum_{\pi \in S_n} \text{sgn}(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}, \tag{4.1}$$

S_n étant l'ensemble de toutes les permutations de $\{1, \ldots, n\}$ et $\text{sgn}(\pi)$ étant la signature de la permutation π (égale à 1 si π s'obtient par un nombre pair de transpositions à partir de la permutation identité, et à -1 sinon).

Proposition 4.3. *Soit une matrice $A \in \mathbb{Q}^{m \times n}$. Alors* $\text{taille}(\det A) \leq 2\,\text{taille}(A)$.

Preuve. Écrivons $a_{ij} = \frac{p_{ij}}{q_{ij}}$, p_{ij} et q_{ij} étant des entiers relativement premiers. On a alors $\det A = \frac{p}{q}$ avec p et q relativement premiers. Alors $|\det A| \leq \prod_{i,j}(|p_{ij}| + 1)$ et $|q| \leq \prod_{i,j} |q_{ij}|$. Nous avons donc $\text{taille}(q) \leq \text{taille}(A)$ et, puisque $|p| = |\det A||q| \leq \prod_{i,j}(|p_{ij}| + 1)|q_{ij}|$,

$$\text{taille}(p) \leq \sum_{i,j} (\text{taille}(p_{ij}) + 1 + \text{taille}(q_{ij})) = \text{taille}(A).$$

□

Cette observation nous permet de prouver :

Théorème 4.4. *Si le PL rationnel* $\max\{cx : Ax \leq b\}$ *a une solution optimale, il a aussi une solution optimale x telle que* $\text{taille}(x) \leq 4n(\text{taille}(A) + \text{taille}(b))$, *la taille de chaque composante étant au plus* $4(\text{taille}(A) + \text{taille}(b))$. *Si $b = \pm e_i$ (e_i vecteur unité), il existe une sous-matrice A' de A et une solution x vérifiant* $\text{taille}(x) \leq 4n\,\text{taille}(A')$.

Preuve. Par le corollaire 3.5, le maximum est atteint sur une face F de $\{x : Ax \leq b\}$. Soit $F' \subseteq F$ une face minimale. Par la proposition 3.9, $F' = \{x : A'x = b'\}$, $A'x \leq b'$ étant un sous-système de $Ax \leq b$. On peut supposer que les lignes de A' sont linéairement indépendantes. Soit A" une sous-matrice de A' induite sur un nombre maximal de colonnes linéairement indépendantes. Alors $x = (A'')^{-1}b'$, complété de composantes valant zéro, est une solution optimale de notre PL. Par la règle de Cramer les composantes de x valent $x_j = \frac{\det A'''}{\det A''}$, A''' s'obtenant en remplaçant dans A'' la j-ième colonne par b'. Par la proposition 4.3, $\text{taille}(x) \leq n + 2n(\text{taille}(A''') + \text{taille}(A'')) \leq 4n(\text{taille}(A'') + \text{taille}(b'))$. Si $b = \pm e_i$, $|\det(A''')|$ est la valeur absolue d'un sous-déterminant de A''. □

La taille du codage d'une face d'un polytope quand on connaît ses sommets peut s'estimer de la manière suivante :

Lemme 4.5. *Soit $P = \{x : Ax \leq b\}$ un polytope rationnel de $\mathbb{R}^n$; s'il existe $T \in \mathbb{N}$ tel que $\text{taille}(x) \leq T$ pour chaque sommet x de P, on peut supposer que chaque inégalité $ax \leq \beta$ du système $Ax \leq b$ vérifie $\text{taille}(a) + \text{taille}(\beta) \leq 75n^2T$.*

Preuve. Supposons que P soit de pleine dimension. Soit $F = \{x \in P : ax = \beta\}$ une facette de P induite par l'inégalité induite par $ax \leq \beta\}$.

Les sommets $y_1, \ldots, y_t$ de F sont aussi des sommets de P, par la proposition 3.6. Soit c la solution de $Mc = e_1$, M étant une matrice $t \times n$ dont la i-ième ligne est $y_i - y_1$ $(i = 2, \ldots, t)$ et dont la première ligne est un vecteur unité linéairement indépendant des autres lignes. Observons que $\text{rang}(M) = n$ (puisque $\dim F = n - 1$). Donc $c^\top = \kappa a$ avec $\kappa \in \mathbb{R} \setminus \{0\}$.

Par le théorème 4.4, $\text{taille}(c) \leq 4n\,\text{taille}(M')$, M' étant une sous-matrice $n \times n$ non singulière de M. Par la proposition 4.2, $\text{taille}(M') \leq 4nT$ et $\text{taille}(c^\top y_1) \leq 2(\text{taille}(c) + \text{taille}(y_1))$. Donc l'inégalité $c^\top x \leq \delta$ (ou $c^\top x \geq \delta$ si $\kappa < 0$), tel que $\delta := c^\top y_1 = \kappa\beta$, vérifie $\text{taille}(c) + \text{taille}(\delta) \leq 3\,\text{taille}(c) + 2T \leq 48n^2T + 2T \leq 50n^2T$. Une inégalité de ce type pour chaque facette fournit une description de P.

Si $P = \emptyset$, le résultat est trivial ; supposons donc que P ne soit pas de pleine dimension et soit non vide. Soit V l'ensemble des sommets de P. Si $s = (s_1, \ldots, s_n) \in \{-1, 1\}^n$, soit P_s l'enveloppe convexe de $V \cup \{x + s_i e_i : x \in V,\ i = 1, \ldots, n\}$. Chaque P_s est de pleine dimension par le théorème 3.31 et la taille de chacun de ses sommets est $T + n$ (voir corollaire 3.32). Par ce qui précède, P_s est décrit par des inégalités de taille au plus $50n^2(T + n) \leq 75n^2T$ (notons que $T \geq 2n$). Puisque $P = \bigcap_{s \in \{-1,1\}^n} P_s$, cela termine la preuve. □

4.2 Fractions continues

Quand nous disons que les nombres rencontrés lors de l'exécution d'un algorithme ne croissent pas trop vite, nous supposons qu'un rationnel est représenté par le quotient $\frac{p}{q}$ de deux entiers p et q relativement premiers entre eux. Cela suppose que nous puissions trouver le plus grand commun diviseur de deux nombres naturels ; c'est ce que fait précisément un des plus anciens algorithmes :

ALGORITHME D'EUCLIDE

Input Deux nombres p et q.

Output Le plus grand commun diviseur d de p et q, c.-à-d. $\frac{p}{d}$ et $\frac{q}{d}$ sont des entiers relativement premiers.

① **While** $p > 0$ et $q > 0$ **do** :
  If $p < q$ **then** $q := q - \lfloor \frac{q}{p} \rfloor p$ **else** $p := p - \lfloor \frac{p}{q} \rfloor q$.

② **Return** $d := \max\{p, q\}$.

Théorème 4.6. *L'*ALGORITHME D'EUCLIDE *répond correctement. Le nombre d'itérations est au plus* $\text{taille}(p) + \text{taille}(q)$.

Preuve. Comme l'ensemble des diviseurs communs de p et q ne change pas durant l'algorithme (jusqu'à ce qu'un des nombres devienne égal à zéro) l'algorithme répond correctement. À chaque itération, p ou q est remplacé par un nombre au moins deux fois plus petit ; il y a au plus $\log p + \log q + 1$ itérations. □

Puisque aucun nombre supérieur à p ou q n'apparaît au cours de l'algorithme, celui-ci est polynomial.

Un algorithme similaire est connu sous le nom d'EXPANSION EN FRACTIONS CONTINUES. Cette méthode est utilisée pour approximer un nombre donné par un nombre rationnel dont le dénominateur n'est pas trop grand. Si x est un nombre réel positif, posons $x_0 := x$ et $x_{i+1} := \frac{1}{x_i - \lfloor x_i \rfloor}$ pour $i = 1, 2, \ldots$ jusqu'à ce que $x_k \in \mathbb{N}$ pour un indice k. Alors

$$x = x_0 = \lfloor x_0 \rfloor + \frac{1}{x_1} = \lfloor x_0 \rfloor + \frac{1}{\lfloor x_1 \rfloor + \frac{1}{x_2}} = \lfloor x_0 \rfloor + \frac{1}{\lfloor x_1 \rfloor + \frac{1}{\lfloor x_2 \rfloor + \frac{1}{x_3}}} = \cdots$$

Montrons que cette suite est finie si et seulement si x est rationnel. Une partie de la démonstration est immédiate et se déduit de l'observation que x_{i+1} est rationnel si et seulement si x_i est rationnel. L'autre partie est également facile : si $x = \frac{p}{q}$, la procédure est équivalente à l'ALGORITHME D'EUCLIDE appliqué à p et q. Cela montre que si $x = \frac{p}{q}$ avec $p, q > 0$ est un nombre rationnel, la suite (finie) $x_1, x_2, \ldots, x_k$ se calcule en temps polynomial. L'algorithme suivant est presque identique à l'ALGORITHME D'EUCLIDE, seul le calcul des nombres g_i et h_i étant différent ; nous allons montrer que la suite $\left(\frac{g_i}{h_i}\right)_{i \in \mathbb{N}}$ converge vers x.

EXPANSION EN FRACTIONS CONTINUES

Input Deux nombres naturels p et q ($x := \frac{p}{q}$).

Output La suite $\left(x_i = \frac{p_i}{q_i}\right)_{i=0,1,\ldots}$ avec $x_0 = \frac{p}{q}$ et $x_{i+1} := \frac{1}{x_i - \lfloor x_i \rfloor}$.

① $i := 0$, $p_0 := p$ et $q_0 := q$.
$g_{-2} := 0$, $g_{-1} := 1$, $h_{-2} := 1$, et $h_{-1} := 0$.

② **While** $q_i \neq 0$ **do** :
$a_i := \lfloor \frac{p_i}{q_i} \rfloor$.
$g_i := a_i g_{i-1} + g_{i-2}$.
$h_i := a_i h_{i-1} + h_{i-2}$.
$q_{i+1} := p_i - a_i q_i$.
$p_{i+1} := q_i$.
$i := i + 1$.

Faisons les observations préliminaires suivantes :

Proposition 4.7. *Les conditions suivantes sont vérifiées à chaque itération i de l'algorithme précédent :*

(a) $a_i \geq 1$ *(sauf éventuellement pour $i = 0$) et* $h_i \geq h_{i-1}$.

(b) $g_{i-1}h_i - g_ih_{i-1} = (-1)^i$.

(c) $\dfrac{p_ig_{i-1} + q_ig_{i-2}}{p_ih_{i-1} + q_ih_{i-2}} = x$.

(d) $\frac{g_i}{h_i} \le x$ *si* i *est pair* $\frac{g_i}{h_i} \ge x$ *si* i *est impair.*

Preuve. (a) est évident ; (b) se démontre facilement par induction : si $i = 0$ nous avons $g_{i-1}h_i - g_ih_{i-1} = g_{-1}h_0 = 1$, et si $i \ge 1$ nous avons

$$\begin{aligned} g_{i-1}h_i - g_ih_{i-1} &= g_{i-1}(a_ih_{i-1} + h_{i-2}) - h_{i-1}(a_ig_{i-1} + g_{i-2}) \\ &= g_{i-1}h_{i-2} - h_{i-1}g_{i-2}. \end{aligned}$$

(c) se montre également par induction : si $i = 0$ nous avons

$$\frac{p_ig_{i-1} + q_ig_{i-2}}{p_ih_{i-1} + q_ih_{i-2}} = \frac{p_i \cdot 1 + 0}{0 + q_i \cdot 1} = x.$$

Si $i \ge 1$ nous avons

$$\begin{aligned} \frac{p_ig_{i-1} + q_ig_{i-2}}{p_ih_{i-1} + q_ih_{i-2}} &= \frac{q_{i-1}(a_{i-1}g_{i-2} + g_{i-3}) + (p_{i-1} - a_{i-1}q_{i-1})g_{i-2}}{q_{i-1}(a_{i-1}h_{i-2} + h_{i-3}) + (p_{i-1} - a_{i-1}q_{i-1})h_{i-2}} \\ &= \frac{q_{i-1}g_{i-3} + p_{i-1}g_{i-2}}{q_{i-1}h_{i-3} + p_{i-1}h_{i-2}}. \end{aligned}$$

Montrons (d). Notons que $\frac{g_{-2}}{h_{-2}} = 0 < x < \infty = \frac{g_{-1}}{h_{-1}}$ et procédons par induction. Cela se déduit facilement du fait que la fonction $f(\alpha) := \frac{\alpha g_{i-1} + g_{i-2}}{\alpha h_{i-1} + h_{i-2}}$ est monotone pour $\alpha > 0$, et que $f(\frac{p_i}{q_i}) = x$ par (c). □

Théorème 4.8. (Khintchine [1956]) *Soit α un rationnel et soit n un naturel ; on peut trouver en temps polynomial (par rapport à* $\mathrm{taille}(n) + \mathrm{taille}(\alpha)$*) un rationnel β ayant un dénominateur au plus égal à n tel que $|\alpha - \beta|$ soit minimum.*

Preuve. Exécutons l'algorithme d'EXPANSION EN FRACTIONS CONTINUES avec $x := \alpha$. Si l'algorithme se termine avec $q_i = 0$ et $h_{i-1} \le n$, on peut poser $\beta = \frac{g_{i-1}}{h_{i-1}} = \alpha$ par la proposition 4.7(c). Sinon soit i le dernier indice tel que $h_i \le n$, et soit t l'entier maximum tel que $th_i + h_{i-1} \le n$ (voir proposition 4.7(a)). Puisque $a_{i+1}h_i + h_{i-1} = h_{i+1} > n$, il vient $t < a_{i+1}$. Prouvons alors que

$$y := \frac{g_i}{h_i} \quad \text{ou} \quad z := \frac{tg_i + g_{i-1}}{th_i + h_{i-1}}$$

est une solution optimale. Les deux nombres ont un dénominateur au plus n.

Si i est pair, alors $y \le x < z$ par la proposition 4.7(d). De même, si i est impair, alors $y \ge x > z$. Montrons que tout nombre rationnel $\frac{p}{q}$ compris entre y et z a un dénominateur supérieur à n.

Observons que

$$|z - y| = \frac{|h_ig_{i-1} - h_{i-1}g_i|}{h_i(th_i + h_{i-1})} = \frac{1}{h_i(th_i + h_{i-1})}$$

(en utilisant la proposition 4.7(b)). D'autre part,

$$|z-y| \;=\; \left|z-\frac{p}{q}\right| + \left|\frac{p}{q}-y\right| \;\geq\; \frac{1}{(th_i+h_{i-1})q} + \frac{1}{h_i q} = \frac{h_{i-1}+(t+1)h_i}{qh_i(th_i+h_{i-1})},$$

et donc $q \geq h_{i-1}+(t+1)h_i > n$. □

La preuve précédente est tirée du livre de Grötschel, Lovász et Schrijver [1988] qui contient également d'importantes généralisations.

4.3 Méthode d'élimination de Gauss

L'algorithme le plus important en algèbre linéaire est connu sous le nom de MÉTHODE D'ÉLIMINATION DE GAUSS. Cette méthode a été appliquée par Gauss mais était connue bien avant (voir Schrijver [1986] pour des références historiques). Cette méthode trouve le rang d'une matrice, calcule un déterminant et résout un système d'équations linéaires. Elle est souvent un sous-programme des algorithmes de PROGRAMMATION LINÉAIRE ; par exemple dans ① de l'ALGORITHME DU SIMPLEXE.

Soit $A \in \mathbb{Q}^{m\times n}$; la MÉTHODE D'ÉLIMINATION DE GAUSS utilise une matrice complétée $Z = (B \quad C) \in \mathbb{Q}^{m\times(n+m)}$; au départ $B = A$ et $C = I$. L'algorithme transforme B en $\left(\begin{smallmatrix} I & R \\ 0 & 0 \end{smallmatrix}\right)$ par les opérations élémentaires suivantes : permutation de lignes et de colonnes, addition d'un multiple d'une ligne à une autre ligne, et multiplication de lignes par des nombres non nuls. À chaque itération C est modifié, de telle sorte que la propriété $C\tilde{A} = B$, où $\tilde{A}$ se déduit de A par permutation de lignes et permutation de colonnes, est toujours vérifiée.

La première partie de l'algorithme, consistant en ② et ③, transforme B en une matrice triangulaire supérieure. Considérons par exemple la matrice Z après deux itérations ; elle a la forme

$$\left(\begin{array}{ccccccc|ccccccc}
z_{11}\neq 0 & z_{12} & z_{13} & \cdot & \cdot & \cdot & z_{1n} & 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\
0 & z_{22}\neq 0 & z_{23} & \cdot & \cdot & \cdot & z_{2n} & z_{2,n+1} & 1 & 0 & \cdot & \cdot & \cdot & 0 \\
0 & 0 & z_{33} & \cdot & \cdot & \cdot & z_{3n} & z_{3,n+1} & z_{3,n+2} & 1 & 0 & \cdot & \cdot & 0 \\
\cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & 0 & & & & \cdot \\
\cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & \cdot & & I & & \cdot \\
\cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & \cdot & & & & 0 \\
0 & 0 & z_{m3} & \cdot & \cdot & \cdot & z_{mn} & z_{m,n+1} & z_{m,n+2} & 0 & \cdot & \cdot & 0 & 1
\end{array}\right).$$

Si $z_{33} \neq 0$, alors l'étape suivante consiste à soustraire la troisième ligne multipliée par $\frac{z_{i3}}{z_{33}}$ de la i-ième ligne, pour $i = 4, \ldots, m$. Si $z_{33} = 0$ nous devons d'abord échanger la troisième ligne avec une autre ligne et (ou) la troisième colonne avec une autre colonne. Notons que quand on échange deux lignes, on doit aussi échanger les deux colonnes correspondantes de C afin de préserver la propriété $C\tilde{A} = B$. Pour connaître $\tilde{A}$ à chaque instant, nous stockons les permutations des lignes et des

colonnes dans les variables $lig(i)$, $i = 1, \ldots, m$ et $col(j)$, $j = 1, \ldots, n$. Alors $\tilde{A} = (A_{lig(i),col(j)})_{i \in \{1,\ldots,m\}, j \in \{1,\ldots,n\}}$.

La seconde partie de l'algorithme, consistant en ④ et ⑤, est plus simple puisqu'on n'a plus besoin d'effectuer des permutations de lignes et de colonnes.

MÉTHODE D'ÉLIMINATION DE GAUSS

Input Une matrice $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$.

Output Son rang r, une sous-matrice non singulière maximale $A' = (a_{lig(i),col(j)})_{i,j \in \{1,\ldots,r\}}$ de A, son déterminant $d = \det A'$, son inverse $(A')^{-1} = (z_{i,n+j})_{i,j \in \{1,\ldots,r\}}$.

① $r := 0$ et $d := 1$.
$z_{ij} := a_{ij}$, $lig(i) := i$ et $col(j) := j$ $(i = 1, \ldots, m, j = 1, \ldots, n)$.
$z_{i,n+j} := 0$ et $z_{i,n+i} := 1$ pour $1 \leq i, j \leq m$, $i \neq j$.

② Soient $p \in \{r+1, \ldots, m\}$ et $q \in \{r+1, \ldots, n\}$ avec $z_{pq} \neq 0$. **If** aucun p et aucun q n'existent, **then go to** ④.
$r := r + 1$.
If $p \neq r$ **then** échanger z_{pj} et z_{rj} $(j = 1, \ldots, n+m)$, échanger $z_{i,n+p}$ et $z_{i,n+r}$ $(i = 1, \ldots, m)$, et échanger $lig(p)$ et $lig(r)$.
If $q \neq r$ **then** échanger z_{iq} et z_{ir} $(i = 1, \ldots, m)$, et échanger $col(q)$ et $col(r)$.

③ $d := d \cdot z_{rr}$.
For $i := r + 1$ **to** m **do :**
$\alpha := \frac{z_{ir}}{z_{rr}}$.
For $j := r$ **to** $n + r$ **do** : $z_{ij} := z_{ij} - \alpha z_{rj}$.
Go to ②.

④ **For** $k := r$ **down to** 2 **do :**
For $i := 1$ **to** $k - 1$ **do :**
$\alpha := \frac{z_{ik}}{z_{kk}}$.
For $j := k$ **to** $n + r$ **do** $z_{ij} := z_{ij} - \alpha z_{kj}$.

⑤ **For** $k := 1$ **to** r **do :**
For $j := 1$ **to** $n + r$ **do** $z_{kj} := \frac{z_{kj}}{z_{kk}}$.

Théorème 4.9. LA MÉTHODE D'ÉLIMINATION DE GAUSS *répond correctement ; elle s'exécute en un temps* $O(mnr)$.

Preuve. Observons d'abord qu'avant chaque étape ② $z_{ii} \neq 0$ pour $i \in \{1, \ldots, r\}$ et $z_{ij} = 0$ pour tout $j \in \{1, \ldots, r\}$ et $i \in \{j+1, \ldots, m\}$. Donc

$$\det\left((z_{ij})_{i,j \in \{1,2,\ldots,r\}}\right) \;=\; z_{11} z_{22} \cdots z_{rr} \;=\; d \;\neq\; 0.$$

Puisque l'addition du multiple d'une ligne à une autre ligne d'une matrice carrée ne change pas la valeur de son déterminant (cette propriété bien connue découle de la définition (4.1)),

$$\det\left((z_{ij})_{i,j\in\{1,2,\dots,r\}}\right) \;=\; \det\left((a_{lig(i),col(j)})_{i,j\in\{1,2,\dots,r\}}\right)$$

à chaque étape précédant ⑤ ; le déterminant d est donc correctement calculé. A' est une sous-matrice $r \times r$ non singulière de A. Puisque le rang de la matrice $(z_{ij})_{i\in\{1,\dots,m\},j\in\{1,\dots,n\}}$ est r à la fin et puisque les opérations n'ont pas modifié le rang, r est aussi le rang de A.

De plus, $\sum_{j=1}^{m} z_{i,n+j}a_{lig(j),col(k)} = z_{ik}$ pour $i \in \{1,\dots,m\}$ et $k \in \{1,\dots,n\}$ (c.-à-d. $C\tilde{A} = B$ avec nos notations précédentes) est toujours vérifié. (Notons que, à chaque étape, $z_{j,n+j} = 1$ et $z_{i,n+j} = 0$ pour $i \neq j$ et $j = r+1,\dots,m$.) Comme $(z_{ij})_{i,j\in\{1,2,\dots,r\}}$ est à la fin la matrice unité, $(A')^{-1}$ est calculé correctement. Le temps de calcul est manifestement $O(rmn + r^2(n+r)) = O(mnr)$. □

Afin de montrer que la MÉTHODE D'ÉLIMINATION DE GAUSS est un algorithme polynomial, montrons que tous les nombres apparaissant au cours du déroulement de l'algorithme sont polynomialement bornés par la taille de l'input.

Théorème 4.10. (Edmonds [1967]) LA MÉTHODE D'ÉLIMINATION DE GAUSS *est un algorithme polynomial. On peut coder chaque nombre apparaissant au cours de l'algorithme par un mot de* $O(m(m+n)\,\text{taille}(A))$ *bits.*

Preuve. Montrons d'abord que tous les nombres générés en ② et ③ valent 0, 1, ou sont des quotients de sous-déterminants de A. Observons d'abord que les coefficients z_{ij} avec $i \leq r$ ou $j \leq r$ ne sont plus modifiés. Les coefficients z_{ij} avec $j > n+r$ valent 0 (si $j \neq n+i$) ou 1 (si $j = n+i$). De plus, pour tout $s \in \{r+1,\dots,m\}$ et $t \in \{r+1,\dots,n+m\}$

$$z_{st} \;=\; \frac{\det\left((z_{ij})_{i\in\{1,2,\dots,r,s\},j\in\{1,2,\dots,r,t\}}\right)}{\det\left((z_{ij})_{i,j\in\{1,2,\dots,r\}}\right)}.$$

(En développant le déterminant $\det\left((z_{ij})_{i\in\{1,2,\dots,r,s\},j\in\{1,2,\dots,r,t\}}\right)$ suivant la dernière ligne, puisque $z_{sj} = 0$ pour tout $s \in \{r+1,\dots,m\}$ et tout $j \in \{1,\dots,r\}$.)

Nous avons remarqué dans la démonstration du théorème 4.9 que

$$\det\left((z_{ij})_{i,j\in\{1,2,\dots,r\}}\right) \;=\; \det\left((a_{lig(i),col(j)})_{i,j\in\{1,2,\dots,r\}}\right),$$

puisque ajouter le multiple d'une ligne à une autre ligne d'une matrice carrée ne change pas la valeur du déterminant. Par le même argument

$$\det\left((z_{ij})_{i\in\{1,2,\dots,r,s\},j\in\{1,2,\dots,r,t\}}\right) \;=\; \det\left((a_{lig(i),col(j)})_{i\in\{1,2,\dots,r,s\},j\in\{1,2,\dots,r,t\}}\right)$$

pour $s \in \{r+1,\dots,m\}$ et $t \in \{r+1,\dots,n\}$. De plus,

$$\det\left((z_{ij})_{i\in\{1,2,\dots,r,s\},j\in\{1,2,\dots,r,n+t\}}\right) \;=\; \det\left((a_{lig(i),col(j)})_{i\in\{1,2,\dots,r,s\}\setminus\{t\},j\in\{1,2,\dots,r\}}\right)$$

pour tout $s \in \{r+1,\dots,m\}$ et $t \in \{1,\dots,r\}$, ce qui se vérifie en développant le déterminant à gauche dans la formule (après ①) par rapport à la colonne $n+t$.

Nous en concluons qu'à toute étape de ② et ③ tous les nombres z_{ij} valent 0, 1, ou sont des quotients de sous-déterminants de A. Donc, par la proposition 4.3, chaque nombre créé en ② et ③ nécessitera un stockage de $O(\text{taille}(A))$ bits.

Observons finalement que ④ est équivalent à l'application de ② et ③, en choisissant p et q convenablement (en inversant l'ordre des r premières lignes et colonnes). Ainsi chaque nombre engendré en ④ nécessitera donc un stockage de $O\left(\text{taille}\left((z_{ij})_{i\in\{1,\ldots,m\},j\in\{1,\ldots,m+n\}}\right)\right)$ bits, c.-à-d. $O(m(m+n)\,\text{taille}(A))$.

Pour avoir une représentation des nombres z_{ij} assez petite, il faut que le numérateur et le dénominateur de chacun de ces nombres soient relativement premiers à toute étape. On réalise cela en appliquant l'ALGORITHME D'EUCLIDE après chaque calcul. On a donc un temps total de calcul polynomial. □

On peut facilement implémenter la MÉTHODE D'ÉLIMINATION DE GAUSS afin de la transformer en un algorithme fortement polynomial (voir exercice 4).

On peut donc en temps polynomial vérifier si un ensemble de vecteurs est linéairement indépendant et on peut calculer le déterminant d'une matrice ou l'inverse d'une matrice carrée non singulière également en temps polynomial (la permutation de deux lignes ou de deux colonnes change simplement le signe du déterminant).

Corollaire 4.11. *Étant donné une matrice $A \in \mathbb{Q}^{m\times n}$ et un vecteur $b \in \mathbb{Q}^m$ on peut en temps polynomial trouver un vecteur $x \in \mathbb{Q}^n$ tel que $Ax = b$ ou certifier qu'un tel vecteur n'existe pas.*

Preuve. Calculons une sous-matrice $A' = (a_{lig(i),col(j)})_{i,j\in\{1,\ldots,r\}}$ non singulière, maximale de A et son inverse $(A')^{-1} = (z_{i,n+j})_{i,j\in\{1,\ldots,r\}}$ par la MÉTHODE D'ÉLIMINATION DE GAUSS. Puis posons $x_{col(j)} := \sum_{k=1}^{r} z_{j,n+k} b_{lig(k)}$ pour $j = 1,\ldots,r$ et $x_k := 0$ pour $k \notin \{col(1),\ldots,col(r)\}$. Pour $i = 1,\ldots r$:

$$\begin{aligned}
\sum_{j=1}^{n} a_{lig(i),j}x_j &= \sum_{j=1}^{r} a_{lig(i),col(j)}x_{col(j)} \\
&= \sum_{j=1}^{r} a_{lig(i),col(j)} \sum_{k=1}^{r} z_{j,n+k} b_{lig(k)} \\
&= \sum_{k=1}^{r} b_{lig(k)} \sum_{j=1}^{r} a_{lig(i),col(j)} z_{j,n+k} \\
&= b_{lig(i)}.
\end{aligned}$$

Les lignes de A dont les indices ne sont pas dans $\{lig(1),\ldots,lig(r)\}$ étant des combinaisons linéaires des autres, x vérifie $Ax = b$ ou ce système n'a pas de solutions. □

4.4 Méthode des ellipsoïdes

Nous présentons dans ce paragraphe la MÉTHODE DES ELLIPSOÏDES, développée par Iudin et Nemirovskii [1976] et Shor [1977] pour l'optimisation non linéaire. Khachiyan [1979] a observé qu'on pouvait l'adapter pour résoudre la programmation linéaire en temps polynomial. L'essentiel de notre présentation est fondée sur (Grötschel, Lovász et Schrijver [1981]), (Bland, Goldfarb et Todd [1981]) et le livre de Grötschel, Lovász et Schrijver [1988], recommandé par ailleurs.

L'idée de la MÉTHODE DES ELLIPSOÏDES est la suivante. Nous recherchons une solution réalisable ou une solution optimale d'un PL ; partons d'un ellipsoïde qui contient *a priori* l'ensemble des solutions (une boule suffisamment grande). À chaque itération k, vérifions si le centre x_k de l'ellipsoïde courant est une solution réalisable. Dans le cas contraire, considérons un hyperplan contenant x_k et laissant dans un même demi-espace toutes les solutions. Ces solutions sont donc contenues dans un demi-ellipsoïde ; nous construisons alors le plus petit ellipsoïde contenant complètement ce demi-ellipsoïde et nous poursuivons cette procédure.

Définition 4.12. *Un* **ellipsoïde** *est un ensemble* $E(A,x) = \{z \in \mathbb{R}^n : (z-x)^\top A^{-1}(z-x) \leq 1\}$ *où* A *est une matrice* $n \times n$ *symétrique définie positive.*

Notons que $B(x,r) := E(r^2 I, x)$ (I étant la matrice unité $n \times n$) est la boule euclidienne ayant x comme centre et r pour rayon.

Le volume d'un ellipsoïde $E(A,x)$ est

$$\text{volume}\,(E(A,x)) \;=\; \sqrt{\det A}\;\text{volume}\,(B(0,1))$$

(voir exercice 7). Étant donné un ellipsoïde $E(A,x)$ et un hyperplan $\{z : az = ax\}$, le plus petit ellipsoïde $E(A',x')$ contenant le demi-ellipsoïde $E' = \{z \in E(A,x) : az \geq ax\}$ est appelé l'ellipsoïde de Löwner-John de E' (voir figure 4.1). Il peut être calculé par les formules suivantes :

$$\begin{aligned} A' &= \frac{n^2}{n^2-1}\left(A - \frac{2}{n+1}bb^\top\right), \\ x' &= x + \frac{1}{n+1}b, \\ b &= \frac{1}{\sqrt{a^\top Aa}}Aa. \end{aligned}$$

Une des difficultés de la MÉTHODE DES ELLIPSOÏDES est la présence de racines carrées nécessaires pour connaître b. Nous devons donc accepter les erreurs d'arrondi en augmentant légèrement le rayon de l'ellipsoïde calculé.

MÉTHODE DES ELLIPSOÏDES	
Input	Un nombre $n \in \mathbb{N}$, $n \geq 2$. Un nombre $N \in \mathbb{N}$. $x_0 \in \mathbb{Q}^n$ et $R \in \mathbb{Q}_+$, $R \geq 2$.
Output	Un ellipsoïde $E(A_N, x_N)$.

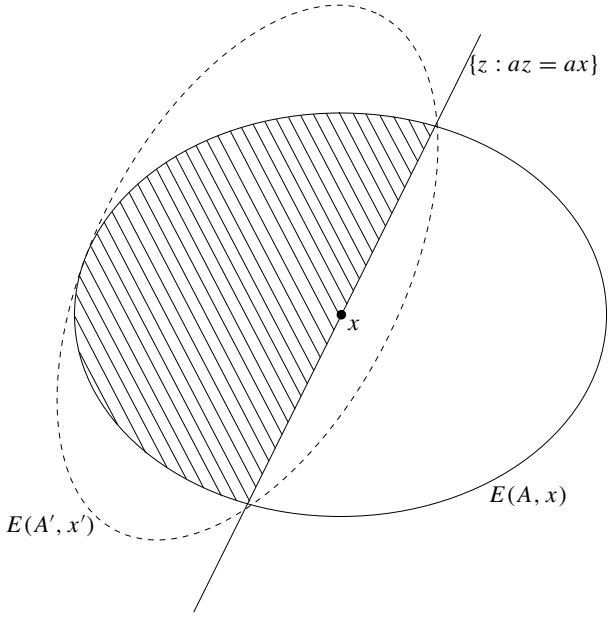


Figure 4.1.

① $p := \lceil 6N + \log(9n^3) \rceil$.
$A_0 := R^2 I$, ou I est la matrice unité $n \times n$.
$k := 0$.

② Choisir $a_k \in \mathbb{Q}^n \setminus \{0\}$.

③ $b_k := \dfrac{1}{\sqrt{a_k^\top A_k a_k}} A_k a_k$.
$x_{k+1} :\approx x^*_{k+1} := x_k + \dfrac{1}{n+1} b_k$.
$A_{k+1} :\approx A^*_{k+1} := \dfrac{2n^2+3}{2n^2}\left(A_k - \dfrac{2}{n+1} b_k b_k^\top\right)$.
(Ici $:\approx$ signifie calculer les coefficients jusqu'à la p-ième décimale, sachant que A_{k+1} est symétrique).

④ $k := k + 1$.
If $k < N$ **then go to** ② **else stop**.

Ainsi, dans chacune des N itérations une approximation $E(A_{k+1}, x_{k+1})$ du plus petit ellipsoïde contenant $E(A_k, x_k) \cap \{z : a_k z \geq a_k x_k\}$ est calculée. Deux points essentiels : comment obtenir les quantités a_k et comment choisir N, seront étudiés au paragraphe suivant. Mais commençons par prouver deux lemmes.

Soit $||x||$ la norme euclidienne du vecteur x et $||A|| := \max\{||Ax|| : ||x|| = 1\}$ la norme de la matrice A. Si A est symétrique, $||A||$ est le maximum de la valeur absolue d'une valeur propre et $||A|| = \max\{x^\top Ax : ||x|| = 1\}$.

Le premier lemme montre que $E_k := E(A_k, x_k)$ est bien un ellipsoïde. De plus, les valeurs absolues des nombres calculés restent inférieures à $R^2 2^N + 2^{\text{taille}(x_0)}$. Il s'ensuit que chaque itération de la MÉTHODE DES ELLIPSOÏDES nécessite $O(n^2)$ étapes de calcul, chacune d'entre elles concernant des nombres de taille $O(p + \text{taille}(a_k) + \text{taille}(R) + \text{taille}(x_0))$.

Lemme 4.13. (Grötschel, Lovász et Schrijver [1981]) *Soit $k \in \{0, 1, \ldots, N\}$. Alors A_k est définie positive, et*

$$||x_k|| \le ||x_0|| + R2^k, \qquad ||A_k|| \le R^2 2^k, \qquad \textit{et} \qquad ||A_k^{-1}|| \le R^{-2} 4^k.$$

Preuve. Par induction sur k. Si $k = 0$ les conditions du lemme sont évidentes. Supposons qu'elles soient vraies pour un entier $k \ge 0$. On peut vérifier directement que

$$(A_{k+1}^*)^{-1} = \frac{2n^2}{2n^2+3}\left(A_k^{-1} + \frac{2}{n-1}\,\frac{a_k a_k^\top}{a_k^\top A_k a_k}\right). \tag{4.2}$$

$(A_{k+1}^*)^{-1}$ qui est la somme d'une matrice définie positive et d'une matrice semi-définie positive est définie positive. Donc A_{k+1}^* est également définie positive.

Notons que si A et B sont semi-définies positives $||A|| \le ||A + B||$. Donc

$$||A_{k+1}^*|| = \frac{2n^2+3}{2n^2}\left|\left|A_k - \frac{2}{n+1} b_k b_k^\top\right|\right| \le \frac{2n^2+3}{2n^2}||A_k|| \le \frac{11}{8} R^2 2^k.$$

Puisque le matrice $n \times n$ dont tous les coefficients valent 1 a une norme égale à n, la matrice $A_{k+1} - A_{k+1}^*$, dont les coefficients ont une valeur absolue égale au plus à 2^{-p}, a une norme égale au plus à $n2^{-p}$. Donc

$$||A_{k+1}|| \le ||A_{k+1}^*|| + ||A_{k+1} - A_{k+1}^*|| \le \frac{11}{8} R^2 2^k + n2^{-p} \le R^2 2^{k+1}$$

(nous utilisons ici l'estimation grossière $2^{-p} \le \frac{1}{n}$).

Soit A une matrice $n \times n$ symétrique définie positive ; il existe une matrice B définie positive telle que $A = BB$. En posant, $A_k = BB$ ($B = B^\top$) nous avons :

$$||b_k|| = \frac{||A_k a_k||}{\sqrt{a_k^\top A_k a_k}} = \sqrt{\frac{a_k^\top A_k^2 a_k}{a_k^\top A_k a_k}} = \sqrt{\frac{(Ba_k)^\top A_k (Ba_k)}{(Ba_k)^\top (Ba_k)}} \le \sqrt{||A_k||} \le R2^{k-1}.$$

Utilisant ce résultat et, de nouveau l'hypothèse d'induction, il vient

$$\begin{aligned} ||x_{k+1}|| &\le ||x_k|| + \frac{1}{n+1}||b_k|| + ||x_{k+1} - x_{k+1}^*|| \\ &\le ||x_0|| + R2^k + \frac{1}{n+1} R2^{k-1} + \sqrt{n}2^{-p} \le ||x_0|| + R2^{k+1}. \end{aligned}$$

Par (4.2) et $||a_k a_k^\top|| = a_k^\top a_k$ nous avons

$$\begin{aligned}
\left|\left|(A_{k+1}^*)^{-1}\right|\right| &\leq \frac{2n^2}{2n^2+3}\left(\left|\left|A_k^{-1}\right|\right| + \frac{2}{n-1}\frac{a_k^\top a_k}{a_k^\top A_k a_k}\right) \qquad (4.3)\\
&= \frac{2n^2}{2n^2+3}\left(\left|\left|A_k^{-1}\right|\right| + \frac{2}{n-1}\frac{a_k^\top B A_k^{-1} B a_k}{a_k^\top B B a_k}\right)\\
&\leq \frac{2n^2}{2n^2+3}\left(\left|\left|A_k^{-1}\right|\right| + \frac{2}{n-1}\left|\left|A_k^{-1}\right|\right|\right) < \frac{n+1}{n-1}\left|\left|A_k^{-1}\right|\right|\\
&\leq 3R^{-2}4^k.
\end{aligned}$$

Soit λ la plus petite valeur propre de A_{k+1}, et soit v le vecteur propre associé à λ avec $||v|| = 1$. Alors – écrivant $A_{k+1}^* = CC$, C étant une matrice symétrique – nous avons

$$\begin{aligned}
\lambda &= v^\top A_{k+1} v = v^\top A_{k+1}^* v + v^\top (A_{k+1} - A_{k+1}^*) v\\
&= \frac{v^\top C C v}{v^\top C \left(A_{k+1}^*\right)^{-1} C v} + v^\top (A_{k+1} - A_{k+1}^*) v\\
&\geq \left|\left|(A_{k+1}^*)^{-1}\right|\right|^{-1} - ||A_{k+1} - A_{k+1}^*|| > \frac{1}{3}R^2 4^{-k} - n2^{-p} \geq R^2 4^{-(k+1)},
\end{aligned}$$

où nous avons utilisé la relation $2^{-p} \leq \frac{1}{3n}4^{-k}$. Puisque $\lambda > 0$, A_{k+1} est définie positive. De plus,

$$\left|\left|(A_{k+1})^{-1}\right|\right| = \frac{1}{\lambda} \leq R^{-2}4^{k+1}.$$

□

Montrons qu'à chaque itération l'ellipsoïde contient l'intersection de E_0 du demi-ellipsoïde précédent :

Lemme 4.14. *$E_{k+1} \supseteq \{x \in E_k \cap E_0 : a_k x \geq a_k x_k\}$ pour $k = 0, \ldots, N-1$.*

Preuve. Soit $x \in E_k \cap E_0$ avec $a_k x \geq a_k x_k$. Utilisant (4.2), calculons d'abord

$$\begin{aligned}
&(x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*)\\
=\ & \frac{2n^2}{2n^2+3}\left(x - x_k - \frac{1}{n+1}b_k\right)^\top \left(A_k^{-1} + \frac{2}{n-1}\frac{a_k a_k^\top}{a_k^\top A_k a_k}\right)\left(x - x_k - \frac{1}{n+1}b_k\right)\\
=\ & \frac{2n^2}{2n^2+3}\Bigg((x-x_k)^\top A_k^{-1}(x-x_k) + \frac{2}{n-1}(x-x_k)^\top \frac{a_k a_k^\top}{a_k^\top A_k a_k}(x-x_k)\\
&\quad + \frac{1}{(n+1)^2}\left(b_k^\top A_k^{-1} b_k + \frac{2}{n-1}\frac{b_k^\top a_k a_k^\top b_k}{a_k^\top A_k a_k}\right)\\
&\quad - \frac{2(x-x_k)^\top}{n+1}\left(A_k^{-1} b_k + \frac{2}{n-1}\frac{a_k a_k^\top b_k}{a_k^\top A_k a_k}\right)\Bigg)\\
=\ & \frac{2n^2}{2n^2+3}\Bigg((x-x_k)^\top A_k^{-1}(x-x_k) + \frac{2}{n-1}(x-x_k)^\top \frac{a_k a_k^\top}{a_k^\top A_k a_k}(x-x_k) +\\
&\quad \frac{1}{(n+1)^2}\left(1 + \frac{2}{n-1}\right) - \frac{2}{n+1}\frac{(x-x_k)^\top a_k}{\sqrt{a_k^\top A_k a_k}}\left(1 + \frac{2}{n-1}\right)\Bigg).
\end{aligned}$$

Puisque $x \in E_k$, $(x - x_k)^\top A_k^{-1}(x - x_k) \leq 1$. En posant $t := \frac{a_k^\top (x - x_k)}{\sqrt{a_k^\top A_k a_k}}$,

$$(x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*) \;\leq\; \frac{2n^2}{2n^2+3}\left(1 + \frac{2}{n-1}t^2 + \frac{1}{n^2-1} - \frac{2}{n-1}t\right).$$

Puisque $b_k^\top A_k^{-1} b_k = 1$ et $b_k^\top A_k^{-1}(x - x_k) = t$,

$$\begin{aligned} 1 &\geq (x - x_k)^\top A_k^{-1}(x - x_k) \\ &= (x - x_k - tb_k)^\top A_k^{-1}(x - x_k - tb_k) + t^2 \\ &\geq t^2, \end{aligned}$$

parce que A_k^{-1} est définie positive. Donc (utilisant la relation $a_k x \geq a_k x_k$) nous obtenons que $0 \leq t \leq 1$ et que

$$(x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*) \;\leq\; \frac{2n^4}{2n^4 + n^2 - 3}.$$

Il reste à estimer l'erreur d'arrondi :

$$\begin{aligned} Z &:= \left|(x - x_{k+1})^\top (A_{k+1})^{-1}(x - x_{k+1}) - (x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*)\right| \\ &\leq \left|(x - x_{k+1})^\top (A_{k+1})^{-1}(x_{k+1}^* - x_{k+1})\right| \\ &\quad + \left|(x_{k+1}^* - x_{k+1})^\top (A_{k+1})^{-1}(x - x_{k+1}^*)\right| \\ &\quad + \left|(x - x_{k+1}^*)^\top \left((A_{k+1})^{-1} - (A_{k+1}^*)^{-1}\right)(x - x_{k+1}^*)\right| \\ &\leq ||x - x_{k+1}||\;||(A_{k+1})^{-1}||\;||x_{k+1}^* - x_{k+1}|| \\ &\quad + ||x_{k+1}^* - x_{k+1}||\;||(A_{k+1})^{-1}||\;||x - x_{k+1}^*|| \\ &\quad + ||x - x_{k+1}^*||^2\;||(A_{k+1})^{-1}||\;||(A_{k+1}^*)^{-1}||\;||A_{k+1}^* - A_{k+1}||. \end{aligned}$$

Utilisant le lemme 4.13 et $x \in E_0$ nous obtenons $||x - x_{k+1}|| \leq ||x - x_0|| + ||x_{k+1} - x_0|| \leq R + R2^N$ et $||x - x_{k+1}^*|| \leq ||x - x_{k+1}|| + \sqrt{n}2^{-p} \leq R2^{N+1}$. En utilisant (4.3),

$$\begin{aligned} Z &\leq 2(R2^{N+1})(R^{-2}4^N)(\sqrt{n}2^{-p}) + (R^2 4^{N+1})(R^{-2}4^N)(3R^{-2}4^{N-1})(n2^{-p}) \\ &= 4R^{-1}2^{3N}\sqrt{n}2^{-p} + 3R^{-2}2^{6N}n2^{-p} \\ &\leq 2^{6N}n2^{-p} \\ &\leq \frac{1}{9n^2}, \end{aligned}$$

par la définition de p. Au total nous obtenons

$$(x - x_{k+1})^\top (A_{k+1})^{-1}(x - x_{k+1}) \;\leq\; \frac{2n^4}{2n^4 + n^2 - 3} + \frac{1}{9n^2} \;\leq\; 1.$$

□

Les volumes des ellipsoïdes décroissent d'un facteur constant à chaque itération :

Lemme 4.15. *Pour* $k = 0, \ldots, N-1$, $\frac{\text{volume}\,(E_{k+1})}{\text{volume}\,(E_k)} < e^{-\frac{1}{5n}}$.

Preuve. (Grötschel, Lovász et Schrijver [1988]) Écrivons

$$\frac{\text{volume}\,(E_{k+1})}{\text{volume}\,(E_k)} \;=\; \sqrt{\frac{\det A_{k+1}}{\det A_k}} \;=\; \sqrt{\frac{\det A^*_{k+1}}{\det A_k}}\,\sqrt{\frac{\det A_{k+1}}{\det A^*_{k+1}}}$$

et estimons les deux facteurs indépendamment. Observons d'abord que

$$\frac{\det A^*_{k+1}}{\det A_k} \;=\; \left(\frac{2n^2+3}{2n^2}\right)^n \det\left(I - \frac{2}{n+1}\,\frac{a_k a_k^\top A_k}{a_k^\top A_k a_k}\right).$$

Le rang de la matrice $\frac{a_k a_k^\top A_k}{a_k^\top A_k a_k}$ est un et 1 est sa seule valeur propre non nulle (a_k vecteur propre associé). Puisque le déterminant est le produit des valeurs propres,

$$\frac{\det A^*_{k+1}}{\det A_k} \;=\; \left(\frac{2n^2+3}{2n^2}\right)^n \left(1 - \frac{2}{n+1}\right) \;<\; e^{\frac{3}{2n}}\, e^{-\frac{2}{n}} \;=\; e^{-\frac{1}{2n}},$$

(par la relation $1 + x \le e^x$ pour tout x et $\left(\frac{n-1}{n+1}\right)^n < e^{-2}$ pour $n \ge 2$).

Pour la seconde estimation nous utiliserons (4.3) et le fait suivant bien connu : $\det B \le ||B||^n$ pour toute matrice B :

$$\begin{aligned}
\frac{\det A_{k+1}}{\det A^*_{k+1}} &= \det\left(I + (A^*_{k+1})^{-1}(A_{k+1} - A^*_{k+1})\right) \\
&\le \left|\left|I + (A^*_{k+1})^{-1}(A_{k+1} - A^*_{k+1})\right|\right|^n \\
&\le \left(||I|| + ||(A^*_{k+1})^{-1}||\;||A_{k+1} - A^*_{k+1}||\right)^n \\
&\le \left(1 + (R^{-2}4^{k+1})(n2^{-p})\right)^n \\
&\le \left(1 + \frac{1}{10n^2}\right)^n \\
&\le e^{\frac{1}{10n}}
\end{aligned}$$

(notons que $2^{-p} \le \frac{4}{10n^3 4^N} \le \frac{R^2}{10n^3 4^{k+1}}$).

En conclusion

$$\frac{\text{volume}\,(E_{k+1})}{\text{volume}\,(E_k)} \;=\; \sqrt{\frac{\det A^*_{k+1}}{\det A_k}}\,\sqrt{\frac{\det A_{k+1}}{\det A^*_{k+1}}} \;\le\; e^{-\frac{1}{4n}}\, e^{\frac{1}{20n}} \;=\; e^{-\frac{1}{5n}}.$$

□

4.5 Théorème de Khachiyan

Dans ce paragraphe nous allons démontrer le théorème de Khachiyan : la MÉTHODE DES ELLIPSOÏDES appliquée à la PROGRAMMATION LINÉAIRE résout ce problème en temps polynomial. Montrons d'abord qu'on peut se ramener au problème de l'existence d'une solution d'un système d'inégalités linéaires :

Proposition 4.16. *S'il existe un algorithme polynomial pour le problème suivant : «étant donné une matrice $A \in \mathbb{Q}^{m\times n}$ et un vecteur $b \in \mathbb{Q}^m$, décider si $\{x : Ax \leq b\}$ est vide», il existe aussi un algorithme polynomial pour la* PROGRAMMATION LINÉAIRE *qui trouve une solution de base si une telle solution existe.*

Preuve. Soit le PL $\max\{cx : Ax \leq b\}$. Vérifions d'abord que le primal et le dual sont réalisables. Si l'un des deux ne l'est pas, la proposition est démontrée par le théorème 3.27. Sinon par le corollaire 3.21, il est suffisant de trouver un élément de $\{(x, y) : Ax \leq b,\ yA = c,\ y \geq 0,\ cx = yb\}$.

Montrons (par induction sur k) qu'on peut trouver une solution d'un système de k inégalités et l égalités par k appels à un sous-programme vérifiant si un polyèdre est vide. Si $k = 0$ on utilise la MÉTHODE D'ÉLIMINATION DE GAUSS (corollaire 4.11).

Si $k > 0$, soit $ax \leq \beta$ une inégalité du système. Par un appel au sous-programme on peut vérifier si le système devient irréalisable quand on remplace $ax \leq \beta$ par $ax = \beta$. Si la réponse est oui alors l'inégalité est redondante et peut être éliminée (proposition 3.8). Si la réponse est non, remplaçons $ax \leq \beta$ par $ax = \beta$. Dans les deux cas, il y a une inégalité de moins, ce qui prouve le résultat par induction.

Une solution optimale de base se trouve par cette procédure puisque le système linéaire final contient un sous-système réalisable maximal de $Ax = b$. □

Avant d'utiliser la MÉTHODE DES ELLIPSOÏDES, nous devons nous assurer que le polyèdre est borné et de pleine dimension :

Proposition 4.17. (Khachiyan [1979], Gács et Lovász [1981]) *Soit $A \in \mathbb{Q}^{m\times n}$ et $b \in \mathbb{Q}^m$. Le système $Ax \leq b$ a une solution si et seulement si le système*

$$Ax \leq b + \epsilon\mathbb{1}, \quad -R\mathbb{1} \leq x \leq R\mathbb{1}$$

a une solution, $\mathbb{1}$ étant le vecteur dont toutes les composantes valent un, $\frac{1}{\epsilon} = 2n2^{4(\text{taille}(A)+\text{taille}(b))}$ et $R = 1 + 2^{4(\text{taille}(A)+\text{taille}(b))}$.

Si $Ax \leq b$ a une solution, alors $\text{volume}\,(\{x \in \mathbb{R}^n : Ax \leq b + \epsilon\mathbb{1}, -R\mathbb{1} \leq x \leq R\mathbb{1}\}) \geq \left(\frac{2\epsilon}{n2^{\text{taille}(A)}}\right)^n$.

Preuve. Les contraintes $-R\mathbb{1} \leq x \leq R\mathbb{1}$ n'altèrent pas la réalisabilité du système $Ax \leq b$ par le théorème 4.4. Si $Ax \leq b$ n'a pas de solution il existe un vecteur $y \geq 0$ vérifiant $yA = 0$ et $yb = -1$ par le théorème 3.24. En appliquant le théorème 4.4 à $\min\{\mathbb{1}y : y \geq 0,\ A^\top y = 0,\ b^\top y = -1\}$ nous voyons qu'on peut choisir y de telle sorte que toutes les valeurs absolues de ses composantes valent au plus $2^{4(\text{taille}(A)+\text{taille}(b))}$. Donc $y(b + \epsilon\mathbb{1}) \leq -1 + n2^{4(\text{taille}(A)+\text{taille}(b))}\epsilon \leq -\frac{1}{2}$. Cela prouve, par le théorème 3.24, que $Ax \leq b + \epsilon\mathbb{1}$ n'a pas de solutions.

Pour la seconde partie, si les valeurs absolues des composantes d'une solution x de $Ax \leq b$ valent au plus $R-1$ (voir théorème 4.4), $\{x \in \mathbb{R}^n : Ax \leq b+\epsilon\mathbb{1}, -R\mathbb{1} \leq x \leq R\mathbb{1}\}$ contient tous les points z tels que $||z - x||_\infty \leq \frac{\epsilon}{n2^{\text{taille}(A)}}$. □

Notons que la construction de la proposition précédente accroît la taille du système d'inégalités par, au plus, un facteur égal à $O(m + n)$.

Théorème 4.18. (Khachiyan [1979]) *Il existe un algorithme pour la* PROGRAMMATION LINÉAIRE *(avec des inputs rationnels) qui trouve une solution de base optimale si une telle solution existe.*

Preuve. Par la proposition 4.16 il suffit de vérifier si le système $Ax \leq b$ est réalisable. En modifiant ce système comme dans la proposition 4.17, nous obtenons un polytope P qui, s'il n'est pas vide, a un volume au moins égal à $\left(\frac{2\epsilon}{n2^{\text{taille}(A)}}\right)^n$. Démarrons la MÉTHODE DES ELLIPSOÏDES avec $R = n\left(1 + 2^{4(\text{taille}(A)+\text{taille}(b))}\right)$, $x_0 = 0$ $N = \lceil 10n^2(2\log n + 5(\text{taille}(A) + \text{taille}(b)))\rceil$. À chaque étape dans ②, vérifions si $x_k \in P$. Dans l'affirmative, nous arrêtons. Sinon, choisissons une inégalité violée $ax \leq \beta$ du système $Ax \leq b$ et posons $a_k := -a$.

Montrons que si l'algorithme ne trouve pas de solution $x_k \in P$ avant l'itération N, alors P est vide. Observons d'abord que $P \subseteq E_k$ pour tout k : cela est vrai si $k = 0$ par la construction de P et R ; c'est vrai également si $k > 0$ par induction en utilisant le lemme 4.14. On a donc $P \subseteq E_N$.

Posons $s := \text{taille}(A) + \text{taille}(b)$; par le lemme 4.15,

$$\begin{aligned} \text{volume}\,(E_N) &\leq \text{volume}\,(E_0)e^{-\frac{N}{5n}} \leq (2R)^n e^{-\frac{N}{5n}} \\ &< \left(2n\left(1+2^{4s}\right)\right)^n n^{-4n} e^{-10ns} < n^{-2n}2^{-5ns}. \end{aligned}$$

D'autre part, $P \neq \emptyset$ implique

$$\text{volume}\,(P) \geq \left(\frac{2\epsilon}{n2^s}\right)^n = \left(\frac{1}{n^2 2^{5s}}\right)^n = n^{-2n}2^{-5ns},$$

ce qui est une contradiction. □

L'estimation du temps de calcul pour résoudre le PL $\max\{cx : Ax \leq b\}$ avec la méthode précédente est $O((n+m)^9(\text{taille}(A)+\text{taille}(b)+\text{taille}(c))^2)$ (voir exercice 9) ; bien que polynomiale, cette borne est sans intérêt pour les applications. En pratique on utilise soit l'ALGORITHME DU SIMPLEXE, soit les algorithmes de points intérieurs. Un algorithme de points intérieurs pour la PROGRAMMATION LINÉAIRE a été décrit pour la première fois par Karmarkar [1984].

On ne connaît aucun algorithme fortement polynomial pour la PROGRAMMATION LINÉAIRE. Cependant, Tardos [1986] a montré qu'il existe un algorithme de résolution de $\max\{cx : Ax \leq b\}$ ayant un temps de calcul qui dépend polynomialement de $\text{taille}(A)$. Pour de nombreux problèmes d'optimisation combinatoire décrits par des matrices 0-1, cela fournit un algorithme fortement polynomial. Le résultat de Tardos a été amélioré par Frank et Tardos [1987].

4.6 Séparation et optimisation

La méthode précédente (en particulier la proposition 4.16) suppose que le polyèdre est explicitement décrit par une liste d'inégalités. Cette condition n'est cependant pas indispensable : il suffit en effet de disposer d'un sous-programme

qui – étant donné un vecteur x – vérifie si $x \in P$ ou alors trouve un hyperplan séparateur, c.-à-d. un vecteur a tel que $ax > \max\{ay : y \in P\}$. Nous supposerons que les polytopes étudiés sont de pleine dimension ; pour le cas général, plus ardu, nous renvoyons à Grötschel, Lovász et Schrijver [1988] (ou Padberg [1995]). Les résultats de cette section sont dus à Grötschel, Lovász et Schrijver [1981] et indépendamment à Karp et Papadimitriou [1982] ainsi qu'à Padberg et Rao [1981].

Grâce aux résultats présentés dans ce paragraphe, nous pourrons résoudre polynomialement des PLs ayant un nombre exponentiel de contraintes induisant des facettes. De nombreux exemples seront discutés plus loin (par exemple le corollaire 12.19 ou le théorème 20.34). En passant à la dualité on pourra aussi résoudre des PLs ayant un très grand nombre de variables.

Soit $P \subseteq \mathbb{R}^n$ un polytope de pleine dimension, ou plus généralement, un ensemble convexe borné de pleine dimension. Nous supposerons que nous connaissons la dimension n et deux boules $B(x_0, r)$ et $B(x_0, R)$ telles que $B(x_0, r) \subseteq P \subseteq B(x_0, R)$. Mais nous ne connaissons pas de système d'inégalités linéaires définissant P. Cette connaissance n'aurait d'ailleurs aucun intérêt s'il s'agissait de résoudre en temps polynomial des PLs ayant un nombre exponentiel de contraintes.

Nous allons montrer que, avec des hypothèses raisonnables, nous pouvons optimiser une forme linéaire sur un polyèdre P en un temps polynomial (indépendant du nombre de contraintes) à l'aide de ce que nous appellerons un **oracle séparateur**, c.-à-d. un sous-programme qui résout le problème suivant :

PROBLÈME DE SÉPARATION

Instance Un ensemble convexe $P \subseteq \mathbb{R}^n$. Un vecteur $y \in \mathbb{Q}^n$.

Tâche Prouver que $y \in P$
ou trouver un vecteur $d \in \mathbb{Q}^n$ tel que $dx < dy$ pour tout $x \in P$.

On notera qu'un tel vecteur d existe si P est un polyèdre rationnel ou un ensemble convexe compact (voir l'exercice 20 du chapitre 3). Si P est un ensemble convexe connu grâce à l'oracle de séparation, nous cherchons un **algorithme fondé sur un oracle** utilisant celui-ci comme une boîte noire : nous pouvons interroger l'oracle séparateur qui nous donne une réponse exacte à tout moment, le temps de chaque appel étant compté comme celui d'une opération élémentaire. (Au chapitre 15 nous donnerons une définition formelle de ce concept.)

Il suffira souvent de disposer d'un oracle séparateur qui résoudra le PROBLÈME DE SÉPARATION approximativement :

PROBLÈME DE SÉPARATION FAIBLE

Instance Un ensemble convexe $P \subseteq \mathbb{R}^n$, un vecteur $c \in \mathbb{Q}^n$ et un nombre $\epsilon > 0$. Un vecteur $y \in \mathbb{Q}^n$.

Tâche Soit trouver un vecteur $y' \in P$ avec $cy \leq cy' + \epsilon$,
soit trouver un vecteur $d \in \mathbb{Q}^n$ tel que $dx < dy$ pour tout $x \in P$.

On résout avec cet oracle séparateur des PLs approchés :

PROBLÈME D'OPTIMISATION FAIBLE

Instance Un nombre $n \in \mathbb{N}$. Un vecteur $c \in \mathbb{Q}^n$. Un nombre $\epsilon > 0$.
Un ensemble convexe $P \subseteq \mathbb{R}^n$ et un oracle pour le PROBLÈME DE SÉPARATION FAIBLE sur P, c et $\frac{\epsilon}{2}$.

Tâche Trouver un vecteur $y \in P$ avec $cy \geq \sup\{cx : x \in P\} - \epsilon$.

Notons que les définitions précédentes diffèrent de celles données dans Grötschel, Lovász et Schrijver [1981] tout en étant équivalentes ; nous aurons besoin de nos définitions au paragraphe 18.3.

La variante suivante de la MÉTHODE DES ELLIPSOÏDES résout le PROBLÈME D'OPTIMISATION FAIBLE pour les ensembles convexes bornés de pleine dimension :

ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER

Input Un nombre $n \in \mathbb{N}$, $n \geq 2$. Un vecteur $c \in \mathbb{Q}^n$. Un nombre $0 < \epsilon \leq 1$.
Un ensemble convexe $P \subseteq \mathbb{R}^n$ donné par un oracle du PROBLÈME DE SÉPARATION FAIBLE pour P, c et $\frac{\epsilon}{2}$.
$x_0 \in \mathbb{Q}^n$ et $r, R \in \mathbb{Q}_+$ tel que $B(x_0, r) \subseteq P \subseteq B(x_0, R)$.

Output Un vecteur $y^* \in P$ avec $cy^* \geq \sup\{cx : x \in P\} - \epsilon$.

① $R := \max\{R, 2\}$, $r := \min\{r, 1\}$ et $\gamma := \max\{||c||, 1\}$.
$N := 5n^2 \left\lceil \ln \frac{6R^2\gamma}{r\epsilon} \right\rceil$. $y^* := x_0$.

② Exécuter la MÉTHODE DES ELLIPSOÏDES, a_k dans ② étant ainsi calculé :
appeler l'oracle du PROBLÈME DE SÉPARATION FAIBLE avec $y = x_k$.
If il renvoie $y' \in P$ avec $cy \leq cy' + \frac{\epsilon}{2}$ **then** :
If $cy' > cy^*$ **then** faire $y^* := y'$.
Faire $a_k := c$.
If il renvoie $d \in \mathbb{Q}^n$ avec $dx < dy$ pour tout $x \in P$ **then** :
Faire $a_k := -d$.

Théorème 4.19. *L'*ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER *résout correctement le* PROBLÈME D'OPTIMISATION FAIBLE *pour les ensembles convexes bornés de pleine dimension. Son temps de calcul est borné par*

$$O\left(n^6\alpha^2 + n^4\alpha f(\text{taille}(c), \text{taille}(\epsilon), n\,\text{taille}(x_0) + n^3\alpha)\right),$$

où $\alpha = \log \frac{R^2\gamma}{r\epsilon}$ *et* $f(\text{taille}(c), \text{taille}(\epsilon), \text{taille}(y))$ *est une borne supérieure du temps de calcul de l'oracle séparateur pour le* PROBLÈME DE SÉPARATION FAIBLE *sur* P *ayant* c, ϵ, y *comme input.*

Preuve. (Grötschel, Lovász et Schrijver [1981]) Le temps de calcul de chacune des $N = O(n^2\alpha)$ itérations de la MÉTHODE DES ELLIPSOÏDES est $O(n^2(n^2\alpha + \text{taille}(R) + \text{taille}(x_0) + q))$ plus un appel à l'oracle, où q est la taille de l'output de l'oracle. Comme $\text{taille}(y) \leq n(\text{taille}(x_0) + \text{taille}(R) + N)$ par le lemme 4.13, le

temps de calcul est $O(n^4\alpha(n^2\alpha + \text{taille}(x_0) + f(\text{taille}(c), \text{taille}(\epsilon), n\,\text{taille}(x_0) + n^3\alpha)))$, comme énoncé.

Par le lemme 4.14,

$$\left\{x \in P : cx \geq cy^* + \frac{\epsilon}{2}\right\} \subseteq E_N.$$

Soit $z \in P$ avec $cz \geq \sup\{cx : x \in P\} - \frac{\epsilon}{6}$. Nous pouvons supposer que $cz > cy^* + \frac{\epsilon}{2}$; sinon l'algorithme est terminé.

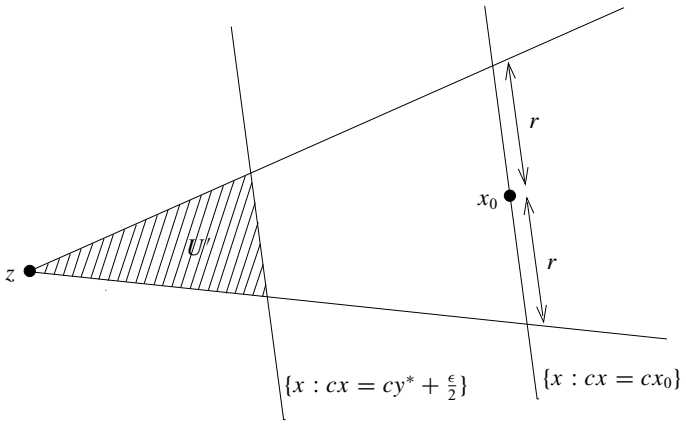


Figure 4.2.

Soit U l'enveloppe convexe de z et de la boule de dimension $(n-1)$ $B(x_0, r) \cap \{x : cx = cx_0\}$ (voir figure 4.2). Comme $U \subseteq P$, $U' := \{x \in U : cx \geq cy^* + \frac{\epsilon}{2}\}$ est contenu dans E_N. Le volume de U' est

$$\begin{aligned}\text{volume}\,(U') &= \text{volume}\,(U)\left(\frac{cz - cy^* - \frac{\epsilon}{2}}{cz - cx_0}\right)^n \\ &= V_{n-1}r^{n-1}\frac{cz - cx_0}{n||c||}\left(\frac{cz - cy^* - \frac{\epsilon}{2}}{cz - cx_0}\right)^n,\end{aligned}$$

où V_n est le volume de la boule unité de dimension n. Puisque $\text{volume}\,(U') \leq \text{volume}\,(E_N)$, le lemme 4.15 s'applique

$$\text{volume}\,(E_N) \;\leq\; e^{-\frac{N}{5n}}\,\text{volume}\,(E_0) \;=\; e^{-\frac{N}{5n}}V_nR^n,$$

et nous avons

$$cz - cy^* - \frac{\epsilon}{2} \;\leq\; e^{-\frac{N}{5n^2}}\,R\left(\frac{V_n(cz - cx_0)^{n-1}n||c||}{V_{n-1}r^{n-1}}\right)^{\frac{1}{n}}.$$

Puisque $cz - cx_0 \leq ||c|| \cdot ||z - x_0|| \leq ||c||R$, :

$$cz - cy^* - \frac{\epsilon}{2} \;\leq\; ||c||e^{-\frac{N}{5n^2}} R \left(\frac{nV_n R^{n-1}}{V_{n-1} r^{n-1}} \right)^{\frac{1}{n}} \;<\; 2||c||e^{-\frac{N}{5n^2}} \frac{R^2}{r} \;\leq\; \frac{\epsilon}{3}$$

et par conséquent $cy^* \geq cz - \frac{5}{6}\epsilon \geq \sup\{cx : x \in P\} - \epsilon$. □

Nous souhaitons cependant obtenir l'optimum exact. Pour cela, nous avons besoin de faire une hypothèse sur la taille des sommets du polytope.

Lemme 4.20. *Soient $n \in \mathbb{N}$, $P \subseteq \mathbb{R}^n$ un polytope rationnel et $x_0 \in \mathbb{Q}^n$ un point à l'intérieur de P. Soit $T \in \mathbb{N}$ tel que* $\text{taille}(x_0) \leq \log T$ *et* $\text{taille}(x) \leq \log T$ *pour tous les sommets x de P. Alors $B(x_0, r) \subseteq P \subseteq B(x_0, R)$, où $r := \frac{1}{n} T^{-379n^2}$ et $R := 2nT$.*

De plus, soit $K := 4T^{2n+1}$. Soit $c \in \mathbb{Z}^n$, et posons $c' := K^n c + (1, K, \ldots, K^{n-1})$. Alors $\max\{c'x : x \in P\}$ est atteint par un vecteur unique x^* ; pour tous les autres sommets y de P, $c'(x^* - y) > T^{-2n}$, et x^* est aussi une solution optimale de $\max\{cx : x \in P\}$.

Preuve. Pour tout sommet x de P nous avons $||x|| \leq nT$ et $||x_0|| \leq nT$, donc $||x - x_0|| \leq 2nT$ et $x \in B(x_0, R)$.

Pour montrer que $B(x_0, r) \subseteq P$, soit $F = \{x \in P : ax = \beta\}$ une facette de P ; par le lemme 4.5 nous pouvons supposer que $\text{taille}(a) + \text{taille}(\beta) < 75n^2 \log T$. S'il existe un point $y \in F$ avec $||y - x_0|| < r$, alors

$$|ax_0 - \beta| \;=\; |ax_0 - ay| \;\leq\; ||a|| \cdot ||y - x_0|| \;<\; n2^{\text{taille}(a)} r \;\leq\; T^{-304n^2}$$

Mais la taille de $ax_0 - \beta$ satisfait l'inégalité

$$\begin{aligned} \text{taille}(ax_0 - \beta) &\leq 4(\text{taille}(a) + \text{taille}(x_0) + \text{taille}(\beta)) \\ &\leq 300n^2 \log T + 4 \log T \;\leq\; 304n^2 \log T. \end{aligned}$$

Puisque $ax_0 \neq \beta$ (x_0 est à l'intérieur de P) $|ax_0 - \beta| \geq T^{-304n^2}$, et on a une contradiction.

Pour montrer le reste du lemme, soit x^* un sommet de P maximisant $c'x$, et soit y un autre sommet de P. Par l'hypothèse sur la taille des sommets de P nous pouvons écrire $x^* - y = \frac{1}{\alpha} z$, où $\alpha \in \{1, 2, \ldots, T^{2n} - 1\}$ et z est un vecteur entier dont les composantes ont une valeur absolue au plus égale à valeur $\frac{K}{2}$. Donc

$$0 \;\leq\; c'(x^* - y) \;=\; \frac{1}{\alpha} \left(K^n cz + \sum_{i=1}^{n} K^{i-1} z_i \right).$$

Puisque $K^n > \sum_{i=1}^{n} K^{i-1}|z_i|$, nous avons nécessairement $cz \geq 0$ et par conséquent $cx^* \geq cy$. Donc x^* maximise bien cx sur le polytope P. De plus, puisque $z \neq 0$, nous obtenons comme souhaité :

$$c'(x^* - y) \;\geq\; \frac{1}{\alpha} \;>\; T^{-2n}.$$

□

Théorème 4.21. *Soit $n \in \mathbb{N}$ et $c \in \mathbb{Q}^n$. Soient $P \subseteq \mathbb{R}^n$ un polytope rationnel et $x_0 \in \mathbb{Q}^n$ un point intérieur à P. Soit $T \in \mathbb{N}$ tel que* $\text{taille}(x_0) \leq \log T$ *et* $\text{taille}(x) \leq \log T$ *pour tous les sommets x de P.*

Étant donné n, c, x_0, T et un oracle polynomial pour le PROBLÈME DE SÉPARATION *sur P, on peut trouver en temps polynomial par rapport à n,* $\log T$ *et* $\text{taille}(c)$ *un sommet x^* de P atteignant* $\max\{c^\top x : x \in P\}$.

Preuve. (Grötschel, Lovász et Schrijver [1981]) Utilisons l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER pour résoudre le PROBLÈME D'OPTIMISATION FAIBLE ; définissons c', r et R comme dans le lemme 4.20 et soit $\epsilon := \frac{1}{8nT^{2n+3}}$. (Nous devons d'abord rendre c entier en le multipliant par le produit des dénominateurs ; cela accroît la taille par un facteur d'au plus $2n$.)

L'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER retourne un vecteur $y \in P$ avec $c'y \geq c'x^* - \epsilon$, x^* étant une solution optimale de $\max\{c'x : x \in P\}$. Le temps de calcul est $O\left(n^6\alpha^2 + n^4\alpha f(\text{taille}(c'), \text{taille}(\epsilon), n\,\text{taille}(x_0) + n^3\alpha)\right) = O\left(n^6\alpha^2 + n^4\alpha f(\text{taille}(c'), 6n\log T, n\log T + n^3\alpha)\right)$ par le théorème 4.19, où $\alpha = \log \frac{R^2 \max\{||c'||,1\}}{r\epsilon} \leq \log(16n^5T^{400n^2}2^{\text{taille}(c')}) = O(n^2 \log T + \text{taille}(c'))$ et f est un polynôme borne supérieure du temps de calcul de l'oracle pour le PROBLÈME DE SÉPARATION pour P. Puisque $\text{taille}(c') \leq 6n^2 \log T + 2\,\text{taille}(c)$, le temps de calcul global est polynomial en n, $\log T$ et $\text{taille}(c)$.

Montrons maintenant que $||x^* - y|| \leq \frac{1}{2T^2}$: écrivons y comme combinaison convexe des sommets $x^*, x_1, \ldots, x_k$ de P :

$$y \;=\; \lambda_0 x^* + \sum_{i=1}^{k} \lambda_i x_i, \quad \lambda_i \geq 0, \quad \sum_{i=0}^{k} \lambda_i = 1.$$

Nous avons – par le lemme 4.20 –

$$\epsilon \;\geq\; c'(x^* - y) \;=\; \sum_{i=1}^{k} \lambda_i c'\,(x^* - x_i) \;>\; \sum_{i=1}^{k} \lambda_i T^{-2n} \;=\; (1-\lambda_0)T^{-2n},$$

et $1 - \lambda_0 < \epsilon T^{2n}$. Donc

$$||y - x^*|| \;\leq\; \sum_{i=1}^{k} \lambda_i ||x_i - x^*|| \;\leq\; (1-\lambda_0)2R \;<\; 4nT^{2n+1}\epsilon \;\leq\; \frac{1}{2T^2}.$$

Quand on arrondit y au rationnel supérieur ayant un dénominateur au plus T, nous obtenons x^*. Cette approximation peut se faire en temps polynomial par le théorème 4.8. □

Nous venons de montrer que, sous certaines hypothèses, optimiser sur un polytope peut se faire grâce à un oracle séparateur. Nous concluons ce chapitre en montrant que le contraire est également vrai. Nous aurons besoin du concept de polarité : si $X \subseteq \mathbb{R}^n$, le **polaire** de X est l'ensemble

$$X^\circ := \{y \in \mathbb{R}^n : y^\top x \leq 1 \text{ pour tout } x \in X\}.$$

La polarité, appliquée aux polytopes de pleine dimension, a les propriétés suivantes :

Théorème 4.22. *Soit P un polytope de $\mathbb{R}^n$ contenant 0 dans son intérieur. Alors :*

(a) *P° est un polytope ayant 0 dans son intérieur.*

(b) *$(P^\circ)^\circ = P$.*

(c) *x est un sommet de P si et seulement si $x^\top y \leq 1$ induit une facette de P°.*

Preuve. (a) : soit P l'enveloppe convexe de $x_1, \ldots, x_k$ (voir théorème 3.31). Par définition, $P^\circ = \{y \in \mathbb{R}^n : y^\top x_i \leq 1 \text{ pour } i \in \{1, \ldots, k\}\}$; P° est donc un polyèdre et les inégalités induisant des facettes de P° sont associées aux sommets de P. De plus, 0 appartient à l'intérieur de P° puisque 0 satisfait strictement l'ensemble fini des inégalités. Si P° est non borné, c.-à-d. s'il existe $w \in \mathbb{R}^n \setminus \{0\}$ avec $\alpha w \in P^\circ$ pour tout $\alpha > 0$, alors $\alpha wx \leq 1$ pour tout $\alpha > 0$ et tout $x \in P$, et donc $wx \leq 0$ pour tout $x \in P$. Mais alors 0 ne peut être dans l'intérieur de P.

(b) : trivialement, $P \subseteq (P^\circ)^\circ$. Pour montrer l'inverse, supposons $z \in (P^\circ)^\circ \setminus P$. Alors il existe une inégalité $c^\top x \leq \delta$ satisfaite par tout $x \in P$ mais pas par z. $\delta > 0$ puisque 0 est dans l'intérieur de P. Alors $\frac{1}{\delta} c \in P^\circ$ mais $\frac{1}{\delta} c^\top z > 1$, ce qui contredit l'hypothèse $z \in (P^\circ)^\circ$.

(c) : nous avons montré en (a) que les inégalités induisant des facettes de P° sont données par les sommets de P. Inversement, si $x_1, \ldots, x_k$ sont les sommets de P, alors $\bar{P} := \text{conv}(\{\frac{1}{2}x_1, x_2, \ldots, x_k\}) \neq P$, et 0 est dans l'intérieur de $\bar{P}$. Mais (b) implique $\bar{P}^\circ \neq P^\circ$. Donc $\{y \in \mathbb{R}^n : y^\top x_1 \leq 2,\ y^\top x_i \leq 1 (i = 2, \ldots, k)\} = \bar{P}^\circ \neq P^\circ = \{y \in \mathbb{R}^n : y^\top x_i \leq 1 (i = 1, \ldots, k)\}$. Nous pouvons alors conclure que $x_1^\top y \leq 1$ est une inégalité induisant une facette de P°. □

Nous pouvons alors démontrer :

Théorème 4.23. *Soit $n \in \mathbb{N}$ et $y \in \mathbb{Q}^n$. Soient $P \subseteq \mathbb{R}^n$ un polytope rationnel et $x_0 \in \mathbb{Q}^n$ un point l'intérieur à P. Soit $T \in \mathbb{N}$ tel que $\text{taille}(x_0) \leq \log T$ et $\text{taille}(x) \leq \log T$ pour tous les sommets x P.*

Étant donné n, y, x_0, T et un oracle qui retourne un sommet x^* de P qui atteint $\max\{c^\top x : x \in P\}$ pour tout $c \in \mathbb{Q}^n$, on peut résoudre le PROBLÈME DE SÉPARATION *pour P et y en temps polynomial par rapport à n, $\log T$ et $\text{taille}(y)$. Si $y \notin P$ on peut trouver une inégalité induisant une facette de P violée par y.*

Preuve. Soient $Q := \{x - x_0 : x \in P\}$ et son polaire Q°. Si $x_1, \ldots, x_k$ sont les sommets de P,

$$Q^\circ = \{z \in \mathbb{R}^n : z^\top (x_i - x_0) \leq 1 \text{ pour tout } i \in \{1, \ldots, k\}\}.$$

Par le théorème 4.4, $\text{taille}(z) \leq 4n(2n \log T + 3n) \leq 20n^2 \log T$ pour tous les sommets z de Q°.

Observons que le PROBLÈME DE SÉPARATION pour P et y est équivalent au PROBLÈME DE SÉPARATION pour Q et $y - x_0$. Puisque par le théorème 4.22

$$Q \;=\; (Q^\circ)^\circ \;=\; \{x : zx \leq 1 \text{ pour tout } z \in Q^\circ\},$$

le PROBLÈME DE SÉPARATION pour Q et $y - x_0$ est équivalent à la résolution de $\max\{(y - x_0)^\top x : x \in Q^\circ\}$. Puisque chaque sommet de Q° correspond à une

inégalité induisant une facette de Q (et donc de P), il suffit de trouver un sommet atteignant $\max\{(y - x_0)^\top x : x \in Q^\circ\}$.

Nous pouvons pour cela appliquer le théorème 4.21 à Q°. Par le théorème 4.22, Q° est de pleine dimension et 0 appartient à son intérieur. Nous avons démontré précédemment que la taille des sommets de Q° est au plus $20n^2 \log T$. Il reste donc à démontrer que nous pouvons résoudre le PROBLÈME DE SÉPARATION pour Q° en temps polynomial. Mais cela se ramène à résoudre le problème d'optimisation pour Q, ce qui se fait en utilisant l'oracle séparateur pour optimiser sur P. □

Mentionnons qu'un nouvel algorithme plus rapide que la MÉTHODE DES ELLIPSOÏDES, et qui implique également l'équivalence entre séparation et optimisation a été proposé par Vaidya [1996]. Cependant, cet algorithme ne semble pas non plus pouvoir être utilisé en pratique.

Exercices

1. Soit A une matrice rationnelle non singulière $n \times n$. Montrer que $\text{taille}(A^{-1}) \leq 4n^2\, \text{taille}(A)$.

* 2. Soit $n \geq 2$, $c \in \mathbb{R}^n$, $y_1, \ldots, y_k \in \{-1, 0, 1\}^n$ tel que $0 < c^\top y_{i+1} \leq \frac{1}{2} c^\top y_i$ pour $i = 1, \ldots, k-1$. Montrer que $k \leq 3n \log n$.
Indication : étudier le PL $\max\{y_1^\top x : y_k^\top x = 1, (y_i - 2y_{i+1})^\top x \geq 0 \ (i = 1, \ldots, k-1)\}$ et revoir la preuve du théorème 4.4.
(M. Goemans)

3. Considérons les nombres h_i dans l'EXPANSION EN FRACTIONS CONTINUES. Montrer que $h_i \geq F_{i+1}$ pour tout i, F_i étant le i-ième nombre de Fibonacci ($F_1 = F_2 = 1$ et $F_n = F_{n-1} + F_{n-2}$ pour $n \geq 3$). Observer que

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right).$$

En conclure que le nombre d'itérations de l'EXPANSION EN FRACTIONS CONTINUES est $O(\log q)$.
(Grötschel, Lovász et Schrijver [1988])

4. Montrer qu'on peut implémenter la méthode d'ÉLIMINATION DE GAUSS pour en faire un algorithme fortement polynomial.
Indication : supposer d'abord que la matrice A est entière. Revoir la preuve du théorème 4.10 et observer qu'on peut choisir d comme étant le plus grand dénominateur commun des coefficients.
(Edmonds [1967])

* 5. Soient $x_1, \ldots, x_k \in \mathbb{R}^l$, $d := 1 + \dim\{x_1, \ldots, x_k\}$, $\lambda_1, \ldots, \lambda_k \in \mathbb{R}_+$ et $\sum_{i=1}^k \lambda_i = 1$, et $x := \sum_{i=1}^k \lambda_i x_i$. Montrer comment calculer les nombres $\mu_1, \ldots, \mu_k \in \mathbb{R}_+$, d d'entre eux au plus étant strictement positifs, de telle sorte que $\sum_{i=1}^k \mu_i = 1$ et $x = \sum_{i=1}^k \mu_i x_i$ (voir exercice 14 du chapitre 3). Montrer

qu'on peut faire tous les calculs en $O((k+l)^3)$.
Indication : exécuter la méthode d'ÉLIMINATION DE GAUSS sur la matrice $A \in \mathbb{R}^{(l+1)\times k}$ ayant pour i-ième colonne $\begin{pmatrix} 1 \\ x_i \end{pmatrix}$. Si $d < k$, soit $w \in \mathbb{R}^k$ le vecteur tel que $w_{col(i)} := z_{i,d+1}$ $(i = 1, \ldots, d)$, $w_{col(d+1)} := -1$ et $w_{col(i)} := 0$ $(i = d+2, \ldots, k)$; observer que $Aw = 0$. Ajouter un multiple de w à λ ; éliminer au moins un vecteur et itérer.

6. Soit $A = \begin{pmatrix} \alpha & b^\top \\ b & C \end{pmatrix} \in \mathbb{R}^{n\times n}$ une matrice symétrique semi-définie positive avec $\alpha > 0$ et $b \in \mathbb{R}^{n-1}$. Soient $A' := \begin{pmatrix} \alpha & 0 \\ 0 & C - \frac{1}{\alpha} bb^\top \end{pmatrix}$ et $U := \begin{pmatrix} 1 & \frac{1}{\alpha} b \\ 0 & I \end{pmatrix}$. Montrer que $A = U^\top A' U$ et $C - \frac{1}{\alpha} bb^\top$ est semi-définie positive. Itérer et conclure que pour toute matrice semi-définie positive A il existe une matrice U telle que $A = U^\top U$, et qu'une telle matrice peut être calculée avec une précision arbitraire en $O(n^3)$ étapes (certaines de ces étapes calculent des approximations de racines carrées).
Note : cela s'appelle la factorisation de Cholesky. Les calculs ne peuvent être exacts, U pouvant être irrationnel.

∗ 7. Soit A une matrice $n \times n$ symétrique définie positive. Soient $v_1, \ldots, v_n$ les n vecteurs propres orthogonaux de A, les valeurs propres associées étant $\lambda_1, \ldots, \lambda_n$. On pourra supposer que $||v_i|| = 1$ pour $i = 1, \ldots, n$. Montrer que

$$E(A,0) = \left\{ \mu_1 \sqrt{\lambda_1} v_1 + \cdots + \mu_n \sqrt{\lambda_n} v_n : \mu \in \mathbb{R}^n,\ ||\mu|| \leq 1 \right\}.$$

(Les vecteurs propres définissent les axes de symétrie de l'ellipsoïde.)
Conclure que $\text{volume}\,(E(A,0)) = \sqrt{\det A}\ \text{volume}\,(B(0,1))$.

8. Soit $E(A,x) \subseteq \mathbb{R}^n$ un ellipsoïde et soit $a \in \mathbb{R}^n$; soit $E(A',x'))$ défini comme en page 83. Montrer que $\{z \in E(A,x) : az \geq ax\} \subseteq E(A',x')$.

9. Montrer que l'algorithme du théorème 4.18 résout un PL $\max\{cx : Ax \leq b\}$ en un temps $O((n+m)^9(\text{taille}(A) + \text{taille}(b) + \text{taille}(c))^2)$.

10. Montrer que l'hypothèse que P est borné n'est pas nécessaire dans le théorème 4.21. On peut détecter si le PL est non borné et sinon, trouver une solution optimale.

∗ 11. Soit $P \subseteq \mathbb{R}^3$ un polytope de dimension 3 ayant 0 dans son intérieur. Soit $G(P)$ le graphe dont les sommets sont les sommets de P et dont les arêtes sont associées aux 1-faces de P (voir exercices 17 et 18 du chapitre 3). Montrer que $G(P^\circ)$ est le dual planaire de $G(P)$.
Note : Steinitz [1922] a montré que, si G est un graphe planaire 3-connexe, il existe un polytope P de dimension 3 tel que $G = G(P)$.

12. Montrer que le polaire d'un polyèdre est toujours un polyèdre. Pour quels polyèdres P a-t-on $(P^\circ)^\circ = P$?

Références

Litterature générale :

Grötschel, M., Lovász, L., Schrijver, A. [1988] : Geometric Algorithms and Combinatorial Optimization. Springer, Berlin 1988

Padberg, M. [1999] : Linear Optimization and Extensions. Second edition. Springer, Berlin 1999

Schrijver, A. [1986] : Theory of Linear and Integer Programming. Wiley, Chichester 1986

Références citées :

Bland, R.G., Goldfarb, D., Todd, M.J. [1981] : The ellipsoïd method : a survey. Operations Research 29 (1981), 1039–1091

Edmonds, J. [1967] : Systems of distinct representatives and linear algebra. Journal of Research of the National Bureau of Standards B 71 (1967), 241–245

Frank, A., Tardos, É. [1987] : An application of simultaneous Diophantine approximation in combinatorial optimization. Combinatorica 7 (1987), 49–65

Gács, P., Lovász, L. [1981] : Khachiyan's algorithm for linear programming. Mathematical Programming Study 14 (1981), 61–68

Grötschel, M., Lovász, L., Schrijver, A. [1981] : The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1 (1981), 169–197

Iudin, D.B., Nemirovskii, A.S. [1976] : Informational complexity and effective methods of solution for convex extremal problems. Ekonomika i Matematicheskie Metody 12 (1976), 357–369 [in Russian]

Karmarkar, N. [1984] : A new polynomial-time algorithm for linear programming. Combinatorica 4 (1984), 373–395

Karp, R.M., Papadimitriou, C.H. [1982] : On linear characterizations of combinatorial optimisation problems. SIAM Journal on Computing 11 (1982), 620–632

Khachiyan, L.G. [1979] : A polynomial algorithm in linear programming [in Russian]. Doklady Akademii Nauk SSSR 244 (1979) 1093–1096. English translation : Soviet Mathematics Doklady 20 (1979), 191–194

Khintchine, A. [1956] : Kettenbrüche. Teubner, Leipzig 1956

Padberg, M.W., Rao, M.R. [1981] : The Russian method for linear programming III : Bounded integer programming. Research Report 81-39, New York University 1981

Shor, N.Z. [1977] : Cut-off method with space extension in convex programming problems. Cybernetics 13 (1977), 94–96

Steinitz, E. [1922] : Polyeder und Raumeinteilungen. Enzyklopädie der Mathematischen Wissenschaften, Band 3 (1922), 1–139

Tardos, É. [1986] : A strongly polynomial algorithm to solve combinatorial linear programs. Operations Research 34 (1986), 250–256

Vaidya, P.M. [1996] : A new algorithm for minimizing convex functions over convex sets. Mathematical Programming 73 (1996), 291–341

Chapitre 5

Programmation en nombres entiers

Le problème de la PROGRAMMATION EN NOMBRES ENTIERS se formule ainsi :

PROGRAMMATION EN NOMBRES ENTIERS	
Instance	Une matrice $A \in \mathbb{Z}^{m\times n}$ et deux vecteurs $b \in \mathbb{Z}^m, c \in \mathbb{Z}^n$.
Tâche	Trouver un vecteur $x \in \mathbb{Z}^n$ tel que $Ax \leq b$ avec cx maximum, montrer que $\{x \in \mathbb{Z}^n : Ax \leq b\} = \emptyset$, ou montrer que $\sup\{cx : x \in \mathbb{Z}^n,\ Ax \leq b\} = \infty$.

Nous n'étudierons pas les PLs mixtes, (PLs avec des contraintes d'intégralité pour un sous-ensemble des variables), mais les résultats de ce chapitre s'étendent naturellement à la PROGRAMMATION LINÉAIRE MIXTE.

Tous les problèmes d'optimisation combinatoire peuvent virtuellement se formuler comme des programmes en nombres entiers. L'ensemble des solutions réalisables est $\{x : Ax \leq b,\ x \in \mathbb{Z}^n\}$, où A est une matrice et b est un vecteur. L'ensemble $P := \{x \in \mathbb{R}^n : Ax \leq b\}$ est un polyèdre ; $P_I = \{x : Ax \leq b\}_I$ sera défini comme l'enveloppe convexe des vecteurs entiers de P. Nous dirons que P_I est l'**enveloppe entière** de P. Bien entendu, $P_I \subseteq P$.

Si P est borné, alors P_I est aussi un polytope par le théorème 3.31 (voir figure 5.1). Meyer [1974] a montré :

Théorème 5.1. *Si P est un polyèdre rationnel, son enveloppe entière P_I est un polyèdre rationnel.*

Preuve. Soit $P = \{x : Ax \leq b\}$. Par le théorème 3.29 le cône polyédral rationnel $C := \{(x, \xi) : x \in \mathbb{R}^n, \xi \geq 0, Ax - \xi b \leq 0\}$ est finiment engendré. On peut supposer que $(x_1, 1), \ldots, (x_k, 1), (y_1, 0), \ldots, (y_l, 0)$ engendrent C ; $x_1, \ldots, x_k$ sont des rationnels et $y_1, \ldots, y_l$ sont des entiers, car on peut multiplier les éléments d'un ensemble fini de générateurs par des scalaires convenablement choisis.

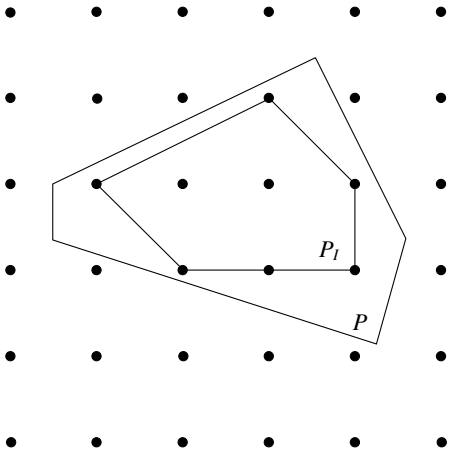


Figure 5.1.

Soit le polytope

$$Q := \left\{ \sum_{i=1}^{k} \kappa_i x_i + \sum_{i=1}^{l} \lambda_i y_i \quad : \quad \kappa_i \geq 0 \,(i = 1, \ldots, k), \right.$$
$$\sum_{i=1}^{k} \kappa_i = 1,$$
$$\left. 0 \leq \lambda_i \leq 1 \,(i = 1, \ldots, l) \right\}.$$

Notons que $Q \subseteq P$. Soient $z_1, \ldots, z_m$ les points entiers de Q. Le cône C' engendré par $(y_1, 0), \ldots, (y_l, 0), (z_1, 1), \ldots, (z_m, 1)$ est polyédral par le théorème 3.29 ; donc $C' = \{(x, \xi) : Mx + \xi b \leq 0\}$, où M est une matrice rationnelle et b est un vecteur rationnel. Nous allons prouver que $P_I = \{x : Mx \leq -b\}$.

Montrons d'abord que $P_I \subseteq \{x : Mx \leq -b\}$. Soit $x \in P \cap \mathbb{Z}^n$. $(x, 1) \in C$ et $x = \sum_{i=1}^{k} \kappa_i x_i + \sum_{i=1}^{l} \lambda_i y_i$ avec $\kappa_1 \ldots, \kappa_k \geq 0$, $\sum_{i=1}^{k} \kappa_i = 1$ et $\lambda_1 \ldots, \lambda_l \geq 0$. $c := \sum_{i=1}^{l} \lfloor \lambda_i \rfloor y_i$ est entier, et $x - c$ est aussi entier. De plus, $x - c = \sum_{i=1}^{k} \kappa_i x_i + \sum_{i=1}^{l} (\lambda_i - \lfloor \lambda_i \rfloor) y_i \in Q$, et $x - c = z_i$, z_i étant un des points entiers de Q. Donc $(x, 1) = (c, 0) + (x - c, 1) \in C'$ et $x \in \{x : Mx \leq -b\}$.

Montrons maintenant que $P_I \supseteq \{x : Mx \leq -b\}$; soit x un vecteur rationnel vérifiant $Mx \leq -b$; $(x, 1) \in C'$ et $x = \sum_{i=1}^{l} \lambda_i y_i + \sum_{i=1}^{m} \mu_i z_j$, avec $\lambda_1 \ldots, \lambda_l, \mu_1, \ldots, \mu_m \geq 0$ et $\sum_{i=1}^{m} \mu_i = 1$. On peut toujours supposer $\mu_1 > 0$. Soit $\delta \in \mathbb{N}$ tel que $\delta \lambda_i \in \mathbb{N}$ pour $i = 1, \ldots, l$ et $\delta \geq \frac{1}{\mu_1}$. Alors $(z_1 + \sum_{i=1}^{l} \delta \lambda_i y_i, 1) \in C$ et

$$x = \frac{1}{\delta} \left(z_1 + \sum_{i=1}^{l} \delta \lambda_i y_i \right) + \left(\mu_1 - \frac{1}{\delta} \right) z_1 + \sum_{i=2}^{m} \mu_i z_i$$

est une combinaison convexe de points entiers de P. □

Ce résultat n'est pas vrai en général pour les polyèdres irrationnels (exercice 1). Par le théorème 5.1, toute instance de la PROGRAMMATION EN NOMBRES ENTIERS peut s'écrire : $\max\{c^\top x : x \in P_I\}$ avec $P = \{x : Ax \leq b\}$.

Nous prouverons une généralisation du théorème de Meyer 5.1 au paragraphe 5.1 (théorème 5.8). Après quelques résultats préliminaires au paragraphe 5.2, nous étudierons sous quelles conditions un polyèdre est entier ($P = P_I$) aux paragraphes 5.3 et 5.4. Un PL en nombres entiers sera dans ce cas équivalent à sa relaxation linéaire (obtenue en négligeant les contraintes d'intégralité) et se résoudra donc en temps polynomial. Nous rencontrerons souvent cette situation dans les chapitres suivants.

En général, la PROGRAMMATION EN NOMBRES ENTIERS est bien plus difficile que la PROGRAMMATION LINÉAIRE, et on ne connaît pas d'algorithmes polynomiaux pour la résoudre. Cela n'est pas surprenant, puisque de nombreux problèmes difficiles peuvent se formuler comme des PLs en nombres entiers. Cependant nous étudierons au paragraphe 5.5 une méthode générale pour trouver l'enveloppe entière en éliminant successivement des parties de $P \setminus P_I$. Cette technique ne conduit pas à un algorithme polynomial, mais peut être utile. Enfin, le paragraphe 5.6 étudie une manière efficace d'approcher la valeur optimale d'un PL en nombres entiers.

5.1 Enveloppe entière d'un polyèdre

Comme les PLs, les programmes en nombres entiers peuvent être non réalisables ou non bornés. Il est difficile de savoir si $P_I = \emptyset$ pour un polyèdre P. Mais si un programme en nombres entiers est réalisable on peut savoir s'il est borné en considérant simplement sa relaxation linéaire.

Proposition 5.2. *Soit $P = \{x : Ax \leq b\}$ un polyèdre rationnel dont l'enveloppe entière est non vide, et soit un vecteur c (pas nécessairement rationnel). Alors $\max\{cx : x \in P\}$ est borné si et seulement si $\max\{cx : x \in P_I\}$ est borné.*

Preuve. Supposons $\max\{cx : x \in P\}$ non borné. Par le corollaire 3.28 le système $yA = c$, $y \geq 0$ n'a pas de solution. Par le corollaire 3.26 il existe un vecteur z tel que $cz < 0$, $Az \geq 0$; le programme $\min\{cz : Az \geq 0, -\mathbb{1} \leq z \leq \mathbb{1}\}$ est donc réalisable. Soit z^* une solution de base optimale de ce PL. z^* est rationnel, car c'est un sommet d'un polytope rationnel. Multiplions z^* par un nombre naturel convenable pour obtenir un vecteur entier w tel que $Aw \geq 0$ et $cw < 0$. Si $v \in P_I$ un vecteur entier, $v - kw \in P_I$ pour $k \in \mathbb{N}$, et $\max\{cx : x \in P_I\}$ est non borné.

L'autre sens de la démonstration est évident. □

Définition 5.3. *Soit A une matrice entière ; si B est une sous-matrice carrée de A, nous dirons que $\det B$ est un* **sous-déterminant** *de A. Nous noterons par $\Xi(A)$ le maximum de la valeur absolue de tous les sous-déterminants de A.*

Lemme 5.4. *Soit $C = \{x : Ax \geq 0\}$ un cône polyédral, A étant une matrice entière. Alors C est engendré par un ensemble fini de vecteurs entiers, chacun d'entre eux ayant des composantes de valeur absolue au plus égales à $\Xi(A)$.*

Preuve. Par le lemme 3.12, C est engendré par des vecteurs $y_1, \ldots, y_t$, solutions de systèmes $My = b'$, M étant une sous-matrice induite sur n lignes linéairement indépendantes de $\binom{A}{I}$ et $b' = \pm e_j$, e_j étant un des vecteurs unité. Soit $z_i := |\det M| y_i$. Par la règle de Cramer, z_i est entier et $||z_i||_\infty \leq \Xi(A)$. Comme cela est vrai pour tout i, l'ensemble $\{z_1, \ldots, z_t\}$ a bien les propriétés souhaitées. □

Un lemme similaire sera utilisé dans le paragraphe suivant :

Lemme 5.5. *Tout cône polyédral rationnel C est engendré par un ensemble de vecteurs entiers $\{a_1, \ldots, a_t\}$, de telle sorte que chaque vecteur entier de C soit combinaison entière non négative de $a_1, \ldots, a_t$. (Cet ensemble est appelé* **base de Hilbert** *de C.)*

Preuve. Supposons que C soit engendré par les vecteurs entiers $b_1, \ldots, b_k$. Nous noterons par $a_1, \ldots, a_t$ les vecteurs entiers du polytope

$$\{\lambda_1 b_1 + \ldots + \lambda_k b_k : 0 \leq \lambda_i \leq 1 \ \ (i = 1, \ldots, k)\}.$$

Montrons que $\{a_1, \ldots, a_t\}$ est une base de Hilbert pour C. Cet ensemble engendre C, puisqu'il inclut l'ensemble $\{b_1, \ldots, b_k\}$.

Si x est un vecteur entier de C, il existe $\mu_1, \ldots, \mu_k \geq 0$ tel que

$$\begin{aligned} x \ = \ \mu_1 b_1 + \ldots + \mu_k b_k \ \ &= \ \ \lfloor \mu_1 \rfloor b_1 + \ldots + \lfloor \mu_k \rfloor b_k + \\ &\quad (\mu_1 - \lfloor \mu_1 \rfloor) b_1 + \ldots + (\mu_k - \lfloor \mu_k \rfloor) b_k, \end{aligned}$$

donc x est combinaison entière non négative de $a_1, \ldots, a_t$. □

Un point important en programmation en nombres entiers concerne la proximité des solutions optimales entières avec les solutions optimales fractionnaires :

Théorème 5.6. (Cook *et al.* [1986]) *Soit A une matrice entière $m \times n$ et $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ deux vecteurs arbitraires. Soit $P := \{x : Ax \leq b\}$ et supposons $P_I \neq \emptyset$.*

(a) *Si y est une solution optimale de $\max\{cx : x \in P\}$, il existe une solution optimale entière z de $\max\{cx : x \in P_I\}$ telle que $||z - y||_\infty \leq n\,\Xi(A)$.*

(b) *Si y est une solution réalisable entière non optimale de $\max\{cx : x \in P_I\}$, il existe une solution réalisable entière $z \in P_I$ telle que $cz > cy$ et $||z - y||_\infty \leq n\,\Xi(A)$.*

Preuve. La preuve est presque identique pour (a) et (b). Soit $y \in P$ arbitraire. On supposera que $z^* \in P \cap \mathbb{Z}^n$ est pour (a) une solution optimale de $\max\{cx : x \in P_I\}$ (notons que $P_I = \{x : Ax \leq \lfloor b \rfloor\}_I$ est un polyèdre par le théorème 5.1 et le maximum est donc atteint) ou pour (b) un vecteur tel que $cz^* > cy$.

Séparons $Ax \leq b$ en deux sous-systèmes $A_1 x \leq b_1$, $A_2 x \leq b_2$ tels que $A_1 z^* \geq A_1 y$ et $A_2 z^* < A_2 y$. $z^* - y$ appartient au cône polyédral $C := \{x : A_1 x \geq 0, A_2 x \leq 0\}$.

C est engendré par des vecteurs x_i $(i = 1, \ldots, s)$. Par le lemme 5.4, nous pouvons supposer que x_i est entier et que $||x_i||_\infty \leq \Xi(A)$ pour tout i.

Puisque $z^* - y \in C$, il existe des nombres non négatifs $\lambda_1, \ldots, \lambda_s$ avec $z^* - y = \sum_{i=1}^{s} \lambda_i x_i$. Nous pouvons supposer qu'au plus n de ces λ_i sont différents de zéro.

Associons à $\mu = (\mu_1, \ldots, \mu_s)$ tel que $0 \leq \mu_i \leq \lambda_i$ $(i = 1, \ldots, s)$ le vecteur

$$z_\mu := z^* - \sum_{i=1}^{s} \mu_i x_i = y + \sum_{i=1}^{s} (\lambda_i - \mu_i) x_i.$$

$z_\mu \in P$, car la première égalité définissant z_μ montre que $A_1 z_\mu \leq A_1 z^* \leq b_1$; et la seconde montre que $A_2 z_\mu \leq A_2 y \leq b_2$.

Cas 1 : il existe $i \in \{1, \ldots, s\}$ tel que $\lambda_i \geq 1$ et $cx_i > 0$. Soit $z := y + x_i$. Alors $cz > cy$; ce cas ne peut donc se produire en (a). Pour (b), quand y est entier, z est une solution entière de $Ax \leq b$ telle que $cz > cy$ et $||z - y||_\infty = ||x_i||_\infty \leq \Xi(A)$.

Cas 2 : pour tout $i \in \{1, \ldots, s\}$, $\lambda_i \geq 1$ implique $cx_i \leq 0$. Soit

$$z \ := z_{\lfloor \lambda \rfloor} \ = \ z^* - \sum_{i=1}^{s} \lfloor \lambda_i \rfloor \, x_i.$$

z est un vecteur entier de P tel que $cz \geq cz^*$ et

$$||z - y||_\infty \ \leq \ \sum_{i=1}^{s} (\lambda_i - \lfloor \lambda_i \rfloor) \, ||x_i||_\infty \ \leq \ n \, \Xi(A).$$

Donc dans les deux cas (a) ou (b) ce vecteur z satisfait le théorème. □

Nous pouvons maintenant borner la taille des solutions optimales des programmes en nombres entiers :

Corollaire 5.7. *Si $P = \{x \in \mathbb{Q}^n : Ax \leq b\}$ est un polyèdre rationnel et $\max\{cx : x \in P_I\}$ a une solution optimale, il existe aussi une solution optimale entière x telle que $\text{taille}(x) \leq 12n(\text{taille}(A) + \text{taille}(b))$.*

Preuve. Par la proposition 5.2 et le théorème 4.4, $\max\{cx : x \in P\}$ a une solution optimale y telle que $\text{taille}(y) \leq 4n(\text{taille}(A) + \text{taille}(b))$. Par le théorème 5.6(a) il existe une solution optimale x de $\max\{cx : x \in P_I\}$ telle que $||x - y||_\infty \leq n \, \Xi(A)$. Par les propositions 4.1 et 4.3

$$\begin{aligned} \text{taille}(x) &\leq 2\,\text{taille}(y) + 2n\,\text{taille}(n\,\Xi(A)) \\ &\leq 8n(\text{taille}(A) + \text{taille}(b)) + 2n \log n + 4n\,\text{taille}(A) \\ &\leq 12n(\text{taille}(A) + \text{taille}(b)). \end{aligned}$$

□

Le théorème 5.6(b) implique le résultat suivant : étant donné une solution réalisable d'un programme en nombres entiers, l'optimalité d'un vecteur x peut être vérifiée en testant $x + y$ pour un nombre fini de vecteurs y qui dépendent de la matrice A uniquement. Un tel ensemble test fini (dont l'existence a été prouvée en premier par Graver [1975]) permet de démontrer un théorème fondamental en PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS :

Théorème 5.8. (Wolsey [1981], Cook *et al.* [1986]) *Si A est une matrice $m \times n$ entière, il existe une matrice entière M dont les coefficients ont une valeur absolue inférieure ou égale à $n^{2n}\Xi(A)^n$, de telle sorte que pour chaque vecteur $b \in \mathbb{Q}^m$ il existe un vecteur rationnel d tel que*

$$\{x : Ax \leq b\}_I \;=\; \{x : Mx \leq d\}.$$

Preuve. Nous pouvons supposer $A \neq 0$. Soit C le cône engendré par les lignes de A et soit

$$L \;:=\; \{z \in \mathbb{Z}^n : ||z||_\infty \leq n\Xi(A)\}.$$

Associons à $K \subseteq L$ le cône

$$C_K \;:=\; C \cap \{y : zy \leq 0 \;\text{ pour tout }\; z \in K\}.$$

Le théorème 3.29 et le lemme 5.4 impliquent que $C_K = \{y : Uy \leq 0\}$, U étant une matrice entière dont les lignes sont les générateurs de $\{x : Ax \leq 0\}$ et les éléments de K ; les coefficients de U ont une valeur absolue inférieure ou égale à $n\Xi(A)$. Donc, de nouveau par le lemme 5.4, il existe un ensemble fini $G(K)$ de vecteurs entiers engendrant C_K ayant des composantes dont la valeur absolue est inférieure ou égale à $\Xi(U) \leq n!(n\Xi(A))^n \leq n^{2n}\Xi(A)^n$.

Soit M la matrice ayant pour lignes $\bigcup_{K \subseteq L} G(K)$. Puisque $C_\emptyset = C$, nous pouvons supposer que les lignes de A sont également des lignes de M.

Soit b un vecteur fixé. Si $Ax \leq b$ n'a pas de solution, nous pouvons compléter b par un vecteur d arbitraire de telle sorte que $\{x : Mx \leq d\} \subseteq \{x : Ax \leq b\} = \emptyset$.

Si $Ax \leq b$ a une solution, mais aucune solution entière, posons $b' := b - A'\mathbb{1}$, A' se déduisant de A en remplaçant chaque coefficient par sa valeur absolue. $Ax \leq b'$ n'a pas de solution, car en prenant les parties entières de chaque composante, on obtiendrait une solution entière de $Ax \leq b$. Nous sommes donc ramenés au cas précédent, en complétant arbitrairement b'.

Supposons que $Ax \leq b$ ait une solution entière. Associons à $y \in C$ la valeur

$$\delta_y \;:=\; \max\{yx : Ax \leq b,\; x \text{ entier}\}$$

(par le corollaire 3.28 le maximum est borné quand $y \in C$). Montrons alors que

$$\{x : Ax \leq b\}_I \;=\; \left\{ x : yx \leq \delta_y \;\text{ pour tout }\; y \in \bigcup_{K \subseteq L} G(K) \right\}. \tag{5.1}$$

Le sens $\subseteq$ est évident. Pour montrer l'autre sens, soit c un vecteur tel que

$$\max\{cx : Ax \leq b,\; x \text{ entier}\}$$

est borné, et soit x^* un vecteur atteignant ce maximum. Montrons que $cx \leq cx^*$ pour tout x satisfaisant les inégalités du membre droit de (5.1).

Par la proposition 5.2 le PL $\max\{cx : Ax \leq b\}$ est borné et $c \in C$ par le corollaire 3.28.

Soit $\bar{K} := \{z \in L : A(x^* + z) \leq b\}$. Par définition $cz \leq 0$ pour tout $z \in \bar{K}$, et $c \in C_{\bar{K}}$. Il existe donc des nombres non négatifs λ_y ($y \in G(\bar{K})$) tels que

$$c \;=\; \sum_{y \in G(\bar{K})} \lambda_y y.$$

Montrons maintenant que x^* est solution optimale de

$$\max \{yx : Ax \leq b,\ x \text{ entier}\}$$

pour tout $y \in G(\bar{K})$: dans le cas contraire, il existerait par le théorème 5.6(b) un vecteur $z \in \bar{K}$ tel que $yz > 0$, ce qui est impossible puisque $y \in C_{\bar{K}}$. En conclusion

$$\sum_{y \in G(\bar{K})} \lambda_y \delta_y \;=\; \sum_{y \in G(\bar{K})} \lambda_y y x^* \;=\; \left(\sum_{y \in G(\bar{K})} \lambda_y y \right) x^* \;=\; cx^*.$$

L'inégalité $cx \leq cx^*$ est donc une combinaison linéaire non négative des inégalités $yx \leq \delta_y$ pour $y \in G(\bar{K})$, ce qui démontre (5.1). □

Voir Lasserre [2004] pour un résultat similaire.

5.2 Transformations unimodulaires

Nous allons montrer dans ce paragraphe deux lemmes qui seront utilisés ultérieurement. Nous dirons qu'une matrice carrée est **unimodulaire** si elle est entière et si son déterminant est égal à $+1$ ou -1. Trois types de matrices unimodulaires auront un intérêt particulier : si $n \in \mathbb{N}$, $p \in \{1, \ldots, n\}$ et $q \in \{1, \ldots, n\} \setminus \{p\}$, ces matrices $(a_{ij})_{i,j \in \{1,\ldots,n\}}$ sont définies d'une des trois manières suivantes :

$$a_{ij} = \begin{cases} 1 & \text{si } i = j \neq p \\ -1 & \text{si } i = j = p \\ 0 & \text{sinon} \end{cases} \qquad a_{ij} = \begin{cases} 1 & \text{si } i = j \notin \{p, q\} \\ 1 & \text{si } \{i, j\} = \{p, q\} \\ 0 & \text{sinon} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{si } i = j \\ -1 & \text{si } (i, j) = (p, q) \\ 0 & \text{sinon} \end{cases}.$$

Ces matrices sont clairement unimodulaires. Si U est l'une d'entre elles, alors remplacer une matrice arbitraire A (ayant n colonnes) par AU revient à appliquer à A une des opérations suivantes sur les colonnes :

- multiplier une colonne par -1 ;
- échanger deux colonnes ;
- soustraire une colonne à une colonne.

Une suite d'opérations de ce type est appelée **transformation unimodulaire**. Le produit de matrices unimodulaires est unimodulaire. On peut montrer qu'une matrice est unimodulaire si et seulement si elle provient d'une matrice identité par une transformation unimodulaire (on dit qu'elle est le produit de matrices d'un des types précédents) ; voir l'exercice 6. Nous n'aurons pas besoin de ce résultat ici.

Proposition 5.9. *L'inverse d'une matrice unimodulaire est unimodulaire. Si U est unimodulaire, les applications $x \mapsto Ux$ et $x \mapsto xU$ sont des bijections sur $\mathbb{Z}^n$.*

Preuve. Soit U une matrice unimodulaire. Par la règle de Cramer, l'inverse d'une matrice unimodulaire est entière. Puisque $(\det U)(\det U^{-1}) = \det(UU^{-1}) = \det I = 1$, U^{-1} est aussi unimodulaire. La seconde condition de l'énoncé découle directement de cela. □

Lemme 5.10. *On peut associer à toute matrice rationnelle A dont les lignes sont linéairement indépendantes une matrice unimodulaire U telle que AU soit égale à $(B \quad 0)$, B étant une matrice carrée non singulière.*

Preuve. Supposons que nous ayons trouvé une matrice unimodulaire U telle que

$$AU = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix}$$

avec B matrice carrée non singulière. (Initialement $U = I$, $D = A$, et les parties B, C, 0 n'existent pas.)

Soit $(\delta_1, \ldots, \delta_k)$ la première ligne de D. Appliquons à D une transformation unimodulaire afin que tous les δ_i soient non négatifs et que $\sum_{i=1}^{k} \delta_i$ soit minimum. On peut supposer $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_k$; $\delta_1 > 0$ puisque les lignes de A (et celles de AU) sont linéairement indépendantes. Si $\delta_2 > 0$, on peut soustraire la seconde colonne de D à la première et faire décroître $\sum_{i=1}^{k} \delta_i$. Donc $\delta_2 = \delta_3 = \ldots = \delta_k = 0$. Nous pouvons ajouter à B une ligne et une colonne et poursuivre. □

Remarquons que les opérations utilisées dans cette preuve correspondent à l'ALGORITHME D'EUCLIDE. La matrice B que nous obtenons est triangulaire inférieure. On peut en allant plus loin obtenir la forme normale d'Hermite de A. Le lemme suivant donne des conditions pour l'existence de solutions entières d'un système linéaire (à rapprocher du lemme de Farkas).

Lemme 5.11. *Soit A une matrice rationnelle et b un vecteur colonne rationnel. Alors, $Ax = b$ a une solution entière si et seulement si yb est un entier pour tout vecteur rationnel y tel que yA est entier.*

Preuve. La condition nécessaire est triviale : si x et yA sont des vecteurs entiers et si $Ax = b$, alors $yb = yAx$ est un entier.

Pour montrer que la condition est suffisante, supposons que yb soit entier quand yA est entier. Cette hypothèse implique que $Ax = b$ ne contient pas d'équation redondante : en effet, dans le cas contraire, il existerait un vecteur $y \neq 0$ tel que

$yA = 0$ et $yb \neq 0$. Mais $y' := \frac{1}{2yb}y$ satisfait $y'A = 0$ et $y'b = \frac{1}{2} \notin \mathbb{Z}$; ce qui contredit l'hypothèse. Les lignes de A sont donc bien linéairement indépendantes.

Par le lemme 5.10 il existe une matrice unimodulaire U telle que $AU = (B \quad 0)$, où B est une matrice $m \times m$ non singulière. Puisque $B^{-1}AU = (I \quad 0)$ est une matrice entière, yAU est entier pour chaque ligne y de B^{-1} et, par la proposition 5.9, yA est entier. Donc yb est entier pour chaque ligne y de B^{-1}, ce qui implique que $B^{-1}b$ est un vecteur entier. $U \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ est une solution entière de $Ax = b$. □

5.3 Totale duale-intégralité

Dans ce paragraphe et dans le suivant nous étudierons les polyèdres entiers :

Définition 5.12. *Un polyèdre P est* **entier** *si $P = P_I$.*

Théorème 5.13. (Hoffman [1974], Edmonds et Giles [1977]) *Soit P un polyèdre rationnel. Les conditions suivantes sont équivalentes :*

(a) *P est entier.*

(b) *Toute face propre de P contient un vecteur entier.*

(c) *Toute face propre minimale de P contient un vecteur entier.*

(d) *Tout hyperplan support de P contient un vecteur entier.*

(e) *Tout hyperplan support rationnel de P contient un vecteur entier.*

(f) *Si $\max\{cx : x \in P\}$ est fini, ce PL a une solution optimale entière.*

(g) *Pour tout c entier, $\max\{cx : x \in P\}$ est entier ou égal à $+\infty$.*

Preuve. Nous montrerons (a)⇒(b)⇒(f)⇒(a), puis (b)⇒(d)⇒(e)⇒(c)⇒(b), et enfin (f)⇒(g)⇒(e).

(a)⇒(b) : soit F une face, par exemple $F = P \cap H$, où H est un hyperplan support, et soit $x \in F$. Si $P = P_I$, x est une combinaison convexe de points entiers de P qui appartiennent à H et donc à F.

(b)⇒(f) grâce à la proposition 3.4, puisque $\{y \in P : cy = \max\{cx : x \in P\}\}$ est une face de P pour tout c pour lequel le maximum est fini.

(f)⇒(a) : supposons qu'il existe un vecteur $y \in P \backslash P_I$. Alors (puisque P_I est un polyèdre par le théorème 5.1) il existe une inégalité $ax \leq \beta$ valide pour P_I telle que $ay > \beta$. Mais alors (f) serait faux, puisque aucun vecteur entier n'atteint $\max\{ax : x \in P\}$ (qui est fini par la proposition 5.2).

(b)⇒(d) est évident puisque l'intersection d'un hyperplan support avec P est une face de P. (d)⇒(e) et (c)⇒(b) sont évidents.

(e)⇒(c) : soit $P = \{x : Ax \leq b\}$. Nous pouvons supposer que A et b sont entiers. Soit $F = \{x : A'x = b'\}$ une face minimale de P induite par le sous-système $A'x \leq b'$ de $Ax \leq b$ (voir proposition 3.9). Si $A'x = b'$ n'a pas de solution entière, il existe par le lemme 5.11 un vecteur rationnel y tel que $c := yA'$ est entier mais $\delta := yb'$ n'est pas entier. On ne change pas cette propriété en ajoutant des

entiers aux composantes de y puisque A' est entière et b' est entier) ; nous pouvons donc supposer que toutes les composantes de y sont positives. $H := \{x : cx = \delta\}$ est un hyperplan rationnel ne contenant aucun vecteur entier.

Mais H est un hyperplan support ; $F \subseteq H$ est évident et il suffit de montrer que $H \cap P \subseteq F$: si $x \in H \cap P$, $yA'x = cx = \delta = yb'$; donc $y(A'x - b') = 0$. Puisque $y > 0$ et $A'x \leq b'$, cela implique $A'x = b'$ et $x \in F$.

(f)⇒(g) est évident et montrons finalement que (g)⇒(e). Soit $H = \{x : cx = \delta\}$ un hyperplan support rationnel de P ; on a donc $\max\{cx : x \in P\} = \delta$. Si H ne contient pas de vecteur entier, alors il existe par le lemme 5.11 un nombre γ tel que γc est entier mais $\gamma\delta \notin \mathbb{Z}$; alors

$$\max\{(|\gamma|c)x : x \in P\} \;=\; |\gamma| \max\{cx : x \in P\} \;=\; |\gamma|\delta \;\notin\; \mathbb{Z},$$

ce qui contredit notre hypoyhèse. □

Voir aussi Gomory [1963], Fulkerson [1971] et Chvátal [1973] pour des résultats antérieurs. Par (a)⇔(b) et le corollaire 3.6, toute face d'un polyèdre entier est entière. L'équivalence entre (f) et (g) du théorème 5.13 a motivé Edmonds et Giles pour l'introduction des systèmes TDI :

Définition 5.14. (Edmonds et Giles [1977]) *Un système $Ax \leq b$ d'inégalités linéaires est appelé* **total dual-intégral (TDI)** *si dans la relation de dualité*

$$\max\{cx : Ax \leq b\} \;=\; \min\{yb : yA = c,\; y \geq 0\}$$

le dual (min) a une solution optimale entière y pour tout vecteur entier c (quand le minimum est fini).

Cette définition fournit un corollaire simple de (g)⇒(a) dans le théorème 5.13 :

Corollaire 5.15. *Soit $Ax \leq b$ un système TDI où A est rationnel et b est entier. Le polyèdre $\{x : Ax \leq b\}$ est entier.* □

Mais la totale duale-intégralité n'est pas une propriété des polyèdres (voir exercice 8). En général, un système TDI contient plus d'inégalités que nécessaire pour décrire le polyèdre. On ne détruit pas la totale duale-intégralité en ajoutant des inégalités valides :

Proposition 5.16. *Si $Ax \leq b$ est TDI et $ax \leq \beta$ est une inégalité valide pour $\{x : Ax \leq b\}$, alors le système $Ax \leq b,\; ax \leq \beta$ est aussi TDI.*

Preuve. Soit c un vecteur entier tel que $\min\{yb + \gamma\beta : yA + \gamma a = c,\; y \geq 0,\; \gamma \geq 0\}$ soit fini. Puisque $ax \leq \beta$ est valide pour $\{x : Ax \leq b\}$,

$$\begin{aligned}
\min\{yb : yA = c,\; y \geq 0\} &= \max\{cx : Ax \leq b\} \\
&= \max\{cx : Ax \leq b,\; ax \leq \beta\} \\
&= \min\{yb + \gamma\beta : yA + \gamma a = c,\; y \geq 0,\; \gamma \geq 0\}.
\end{aligned}$$

Le premier minimum est atteint par un vecteur entier y^* ; $y = y^*,\; \gamma = 0$ est donc une solution optimale entière pour le second minimum. □

Théorème 5.17. (Giles et Pulleyblank [1979]) *Si P est un polyèdre rationnel, il existe un système TDI $Ax \leq b$ avec A entier tel que $P = \{x : Ax \leq b\}$. On peut choisir b entier si et seulement si P est entier.*

Preuve. Soit $P = \{x : Cx \leq d\}$ avec C et d entiers. On peut supposer que $P \neq \emptyset$. Si F est une face de P, soit

$$K_F := \{c : cz = \max\{cx : x \in P\} \text{ pour tout } z \in F\}.$$

Par le corollaire 3.22 et le théorème 3.29, K_F est un cône rationnel polyédral. Par le lemme 5.5, il existe une base de Hilbert entière $a_1, \ldots, a_t$ engendrant K_F. Soit $\mathcal{S}_F$ le système d'inégalités

$$a_1 x \leq \max\{a_1 x : x \in P\}, \ldots, a_t x \leq \max\{a_t x : x \in P\}.$$

Soit $Ax \leq b$ la réunion de tous ces systèmes $\mathcal{S}_F$ (pour toutes les faces minimales F). Notons que si P est entier, alors b est entier. De plus, $P \subseteq \{x : Ax \leq b\}$.

Soit c un vecteur entier pour lequel $\max\{cx : x \in P\}$ est fini. L'ensemble de vecteurs atteignant ce maximum est une face de P ; soit F une face minimale telle que $cz = \max\{cx : x \in P\}$ pour tout $z \in F$. Soit $\mathcal{S}_F$ le système $a_1 x \leq \beta_1, \ldots, a_t x \leq \beta_t$. Alors $c = \lambda_1 a_1 + \cdots + \lambda_t a_t$, $\lambda_1, \ldots, \lambda_t$ étant des entiers non négatifs. En ajoutant des composantes égales à zéro à $\lambda_1, \ldots, \lambda_t$ on obtient un vecteur entier $\bar{\lambda} \geq 0$ qui vérifie $\bar{\lambda} A = c$; donc $cx = (\bar{\lambda} A)x = \bar{\lambda}(Ax) \leq \bar{\lambda} b = \bar{\lambda}(Az) = (\bar{\lambda} A)z = cz$ pour tout x tel que $Ax \leq b$ et pour tout $z \in F$.

En faisant cela pour chaque ligne c de C, on voit que $x \in P$ si x vérifie $Ax \leq b$. Donc $P = \{x : Ax \leq b\}$. De plus, pour c quelconque, $\bar{\lambda}$ est une solution optimale du dual $\min\{yb : y \geq 0,\ yA = c\}$. Donc $Ax \leq b$ est TDI.

Si P est entier, b est entier. Inversement, si on peut choisir b entier, P est entier par le corollaire 5.15. □

Pour les polyèdres de pleine dimension, il existe un système TDI minimal unique décrivant ces polyèdres (Schrijver [1981]). Prouvons pour un usage ultérieur que toute «face» d'un système TDI est encore TDI :

Théorème 5.18. (Cook [1983]) *Soit $Ax \leq b,\ ax \leq \beta$ un système TDI, où a est entier. Alors le système $Ax \leq b,\ ax = \beta$ est aussi TDI.*

Preuve. (Schrijver [1986]) Soit c un vecteur entier tel que

$$\begin{aligned} & \max\{cx : Ax \leq b,\ ax = \beta\} \\ = \ & \min\{yb + (\lambda - \mu)\beta : y, \lambda, \mu \geq 0,\ yA + (\lambda - \mu)a = c\} \end{aligned} \tag{5.2}$$

est fini. Supposons que $x^*, y^*, \lambda^*, \mu^*$ atteignent ces optima. Posons $c' := c + \lceil \mu^* \rceil a$ et observons que

$$\max\{c'x : Ax \leq b,\ ax \leq \beta\} = \min\{yb + \lambda\beta : y, \lambda \geq 0,\ yA + \lambda a = c'\} \tag{5.3}$$

est fini, parce que $x := x^*$ est réalisable pour le maximum et $y := y^*,\ \lambda := \lambda^* + \lceil \mu^* \rceil - \mu^*$ est réalisable pour le minimum.

Puisque $Ax \leq b,\ ax \leq \beta$ est TDI, le minimum dans (5.3) a une solution optimale entière $\tilde{y}, \tilde{\lambda}$. Posons finalement $y := \tilde{y},\ \lambda := \tilde{\lambda}, \mu := \lceil \mu^* \rceil$ et prouvons que (y, λ, μ) est une solution optimale entière pour le minimum dans (5.2).

Il est clair que (y, λ, μ) est réalisable pour le minimum dans (5.2). De plus,

$$\begin{aligned} yb + (\lambda - \mu)\beta &= \tilde{y}b + \tilde{\lambda}\beta - \lceil \mu^* \rceil \beta \\ &\leq y^*b + (\lambda^* + \lceil \mu^* \rceil - \mu^*)\beta - \lceil \mu^* \rceil \beta \end{aligned}$$

puisque $(y^*, \lambda^* + \lceil \mu^* \rceil - \mu^*)$ est réalisable pour le minimum dans (5.3), et $(\tilde{y}, \tilde{\lambda})$ est une solution optimale. En conclusion,

$$yb + (\lambda - \mu)\beta \ \leq\ y^*b + (\lambda^* - \mu^*)\beta,$$

ce qui montre que (y, λ, μ) est une solution optimale entière pour le minimum dans (5.2). □

Les propriétés suivantes sont des conséquences directes de la définition des systèmes TDI : un système $Ax = b,\ x \geq 0$ est TDI si $\min\{yb : yA \geq c\}$ a une solution optimale entière y pour tout vecteur entier c, quand la valeur du PL est finie. Un système $Ax \leq b,\ x \geq 0$ est TDI si $\min\{yb : yA \geq c,\ y \geq 0\}$ a une solution optimale entière y pour tout vecteur entier c, quand la valeur du PL est finie. On peut se demander s'il existe des matrices A telles que $Ax \leq b,\ x \geq 0$ est TDI pour tout vecteur entier b. Ces matrices forment précisément la classe des matrices totalement unimodulaires.

5.4 Matrices totalement unimodulaires

Définition 5.19. *Une matrice A est* **totalement unimodulaire** *si tout sous-déterminant de A vaut 0, $+1$, ou -1.*

En particulier, tout coefficient d'une matrice totalement unimodulaire est égal à 0, $+1$, ou -1. Le principal résultat de ce paragraphe est :

Théorème 5.20. (Hoffman et Kruskal [1956]) *Une matrice entière A est totalement unimodulaire si et seulement si le polyèdre $\{x : Ax \leq b,\ x \geq 0\}$ est entier quel que soit le vecteur entier b.*

Preuve. Soit A une matrice $m \times n$ et $P := \{x : Ax \leq b,\ x \geq 0\}$. Les faces minimales de P sont les sommets de P.

Supposons d'abord A totalement unimodulaire. Soit b un vecteur entier et x un sommet de P. x est solution de $A'x = b'$, $A'x \leq b'$ étant un sous-système de $\begin{pmatrix} A \\ -I \end{pmatrix} x \leq \begin{pmatrix} b \\ 0 \end{pmatrix}$, et A' étant une matrice $n \times n$ non singulière. Puisque A est totalement unimodulaire, $|\det A'| = 1$, et $x = (A')^{-1}b'$ est entier, par la règle de Cramer.

Supposons maintenant que les sommets de P soient entiers, quel que soit le vecteur entier b. Soit A' une sous-matrice $k \times k$ de A non singulière. Montrons que $|\det A'| = 1$. On peut toujours supposer que A' est indicée par les k premières lignes et les k premières colonnes de A.

Soit B la matrice $m \times m$ indicée sur les k premières et les $m - k$ dernières colonnes de $(A \quad I)$ (voir figure 5.2). On a $|\det B| = |\det A'|$.

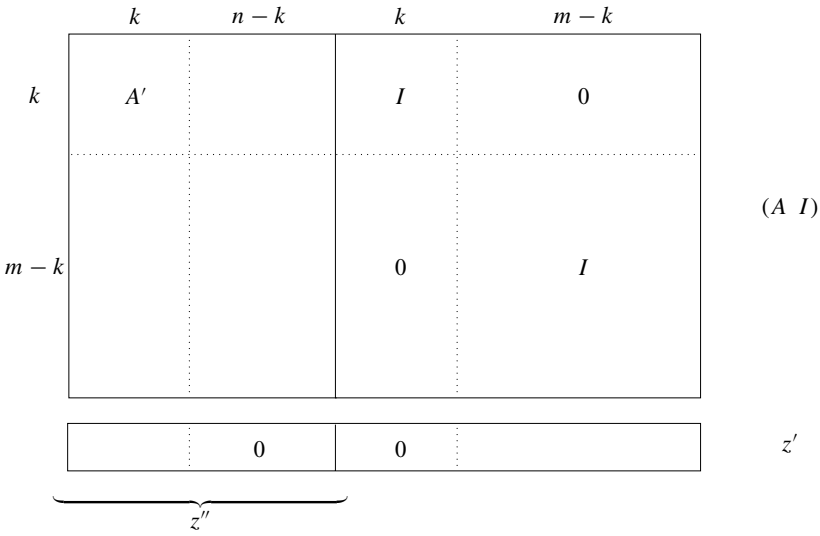


Figure 5.2.

Montrons que la matrice B^{-1} est entière, ce qui entraînera $|\det B| = 1$ puisque $\det B \det B^{-1} = 1$.

Soit $i \in \{1, \ldots, m\}$; montrons que $B^{-1}e_i$ est entier. Choisissons un vecteur entier y tel que $z := y + B^{-1}e_i \geq 0$. $b := Bz = By + e_i$ est donc entier. Ajoutons des composantes nulles à z afin d'obtenir le vecteur z' tel que

$$\begin{pmatrix} A & I \end{pmatrix} z' \;=\; Bz \;=\; b.$$

Le vecteur z'', restriction de z' à ses n premières composantes, appartient à P. De plus z'' satisfait avec égalité n contraintes linéairement indépendantes, à savoir les k premières et les $n - k$ dernières inégalités de

$$\begin{pmatrix} A \\ -I \end{pmatrix} z'' \leq \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Donc z'' est un sommet de P. Par notre hypothèse z'' est entier. z' est donc également entier : ses n premières composantes sont les composantes de z'', et ses

m dernières sont les variables d'écart $b - Az''$ (A est entière et b est entier). Donc z est entier, et $B^{-1}e_i = z - y$ est entier. □

La preuve précédente a été proposée par Veinott et Dantzig [1968].

Corollaire 5.21. *Une matrice entière A est totalement unimodulaire si et seulement si les PLs primal et dual :*

$$\max\{cx : Ax \leq b,\ x \geq 0\} \ = \ \min\{yb : y \geq 0,\ yA \geq c\}$$

ont des solutions optimales entières (quand les deux programmes sont réalisables), quels que soient les vecteurs entiers b et c.

Preuve. C'est une conséquence du théorème de Hoffman-Kruskal 5.20 en remarquant que la transposée d'une matrice totalement unimodulaire est aussi totalement unimodulaire. □

Reformulons ces résultats en termes de totale duale-intégralité :

Corollaire 5.22. *Une matrice entière A est totalement unimodulaire si et seulement si le système $Ax \leq b,\ x \geq 0$ est TDI pour tout vecteur b.*

Preuve. Si A (et donc $A^\top$) est totalement unimodulaire, $\min\{yb : yA \geq c,\ y \geq 0\}$ est atteint par un vecteur entier quels que soient les vecteurs entiers b et c, pourvu que le minimum soit fini, par le théorème de Hoffman-Kruskal. En d'autres termes, le système $Ax \leq b,\ x \geq 0$ est TDI pour tout vecteur b.

Pour montrer le contraire, supposons que $Ax \leq b,\ x \geq 0$ soit TDI quel que soit le vecteur b. Par le corollaire 5.15, le polyèdre $\{x : Ax \leq b,\ x \geq 0\}$ est entier. A est donc totalement unimodulaire par le théorème 5.20. □

Il y a d'autres manières d'utiliser la totale unimodularité pour prouver que certains systèmes sont TDI. Le lemme suivant fournit une autre technique de preuve qui sera souvent utilisée (théorèmes 6.13, 19.10 et 14.12).

Lemme 5.23. *Soit $Ax \leq b,\ x \geq 0$ un système d'inégalités, avec $A \in \mathbb{R}^{m\times n}$ et $b \in \mathbb{R}^m$. Supposons que pour tout $c \in \mathbb{Z}^n$ tel que $\min\{yb : yA \geq c,\ y \geq 0\}$ a une solution optimale, il en existe une y^* dont les indices des composantes non nulles induisent une matrice totalement unimodulaire. Alors $Ax \leq b,\ x \geq 0$ est TDI.*

Preuve. Soit $c \in \mathbb{Z}^n$, et soit y^* une solution optimale de $\min\{yb : yA \geq c,\ y \geq 0\}$ telle que les lignes de A associées aux composantes non nulles de y^* induisent une matrice totalement unimodulaire A'. Montrons que

$$\min\{yb : yA \geq c, y \geq 0\} \ = \ \min\{yb' : yA' \geq c, y \geq 0\}, \tag{5.4}$$

b' étant la restriction de b aux lignes de A'. Pour montrer le sens $\leq$ dans (5.4), observons que le PL du membre droit provient de celui du membre gauche en fixant certaines variables à zéro. Pour l'autre sens $\geq$ remarquons que la restriction de y^* à

ses composantes strictement positives est une solution réalisable du PL du membre droit.

Puisque A' est totalement unimodulaire, le second minimum dans (5.4) a une solution entière optimale (par le théorème de Hoffman-Kruskal 5.20). En complétant cette solution avec des zéros, nous obtenons une solution optimale entière du membre droit de (5.4), ce qui complète la preuve. □

Un critère très utile pour la totale unimodularité est le suivant :

Théorème 5.24. (Ghouila-Houri [1962]) *Une matrice $A = (a_{ij}) \in \mathbb{Z}^{m\times n}$ est totalement unimodulaire si et seulement si quel que soit $R \subseteq \{1,\dots,m\}$ il existe une partition $R = R_1 \dot\cup R_2$ telle que*

$$\sum_{i\in R_1} a_{ij} - \sum_{i\in R_2} a_{ij} \ \in \ \{-1,0,1\}$$

pour tout $j = 1,\dots,n$.

Preuve. Supposons A totalement unimodulaire et soit $R \subseteq \{1,\dots,m\}$. Soit $d_r := 1$ pour $r \in R$ et $d_r := 0$ pour $r \in \{1,\dots,m\} \setminus R$. La matrice $\begin{pmatrix} A^\top \\ -A^\top \\ I \end{pmatrix}$ est totalement unimodulaire et par le théorème 5.20 le polytope

$$\left\{ x : xA \le \left\lceil \frac{1}{2} dA \right\rceil,\ xA \ge \left\lfloor \frac{1}{2} dA \right\rfloor,\ x \le d,\ x \ge 0 \right\}$$

est entier. De plus, il est non vide puisqu'il contient $\frac{1}{2}d$. Il a donc un sommet entier z. Posons $R_1 := \{r \in R : z_r = 0\}$ et $R_2 := \{r \in R : z_r = 1\}$; alors

$$\left(\sum_{i\in R_1} a_{ij} - \sum_{i\in R_2} a_{ij} \right)_{1\le j\le n} = (d-2z)A \ \in \ \{-1,0,1\}^n.$$

Pour l'inverse montrons, par induction sur k, que toute sous-matrice $k \times k$ a un déterminant égal à 0, 1 ou -1. Pour $k = 1$, il suffit de prendre $|R| = 1$.

Supposons alors $k > 1$, et soit $B = (b_{ij})_{i,j\in\{1,\dots,k\}}$ une sous-matrice $k \times k$ non singulière de A. Par la règle de Cramer chaque coefficient de B^{-1} est égal à $\frac{\det B'}{\det B}$, où B' se déduit de B en remplaçant une colonne par un vecteur unité. Par l'hypothèse d'induction, $\det B' \in \{-1,0,1\}$. Donc $B^* := (\det B)B^{-1}$ est une matrice dont les coefficients valent -1, 0 ou 1.

Soit b_1^* la première ligne de B^*. $b_1^* B = (\det B)e_1$, où e_1 est le premier vecteur unité. Soit $R := \{i : b_{1i}^* \neq 0\}$. Pour $j = 2,\dots,k$ nous avons $0 = (b_1^* B)_j = \sum_{i\in R} b_{1i}^* b_{ij}$ et donc $|\{i \in R : b_{ij} \neq 0\}|$ est pair.

Il existe par hypothèse une partition $R = R_1 \dot\cup R_2$ telle que $\sum_{i\in R_1} b_{ij} - \sum_{i\in R_2} b_{ij} \in \{-1,0,1\}$ pour tout j. Donc $\sum_{i\in R_1} b_{ij} - \sum_{i\in R_2} b_{ij} = 0$ pour $j = 2,\dots,k$. Si de plus $\sum_{i\in R_1} b_{i1} - \sum_{i\in R_2} b_{i1} = 0$, la somme des lignes de R_1 est égale à la somme des lignes de R_2, ce qui est impossible puisque B est non singulière (parce que $R \neq \emptyset$).

Donc $\sum_{i\in R_1} b_{i1} - \sum_{i\in R_2} b_{i1} \in \{-1, 1\}$ et $yB \in \{e_1, -e_1\}$, où

$$y_i := \begin{cases} 1 & \text{si } i \in R_1 \\ -1 & \text{si } i \in R_2 \\ 0 & \text{si } i \notin R \end{cases}.$$

Puisque $b_1^* B = (\det B)e_1$ et B est non singulière, $b_1^* \in \{(\det B)y, -(\det B)y\}$. y et b_1^* sont des vecteurs non nuls dont les composantes valent -1, 0 ou 1 et par conséquent $|\det B| = 1$. □

Appliquons ce critère aux matrices d'incidence des graphes :

Théorème 5.25. *La matrice d'incidence d'un graphe non orienté G est totalement unimodulaire si et seulement si G est biparti.*

Preuve. Par le théorème 5.24, la matrice d'incidence M de G est totalement unimodulaire si et seulement si pour tout $X \subseteq V(G)$ il existe une partition $X = A \dot\cup B$ telle que $E(G[A]) = E(G[B]) = \emptyset$. Par définition, une telle partition existe si et seulement si $G[X]$ est biparti. □

Théorème 5.26. *La matrice d'incidence d'un graphe orienté est totalement unimodulaire.*

Preuve. Par le théorème 5.24, si $R \subseteq V(G)$, il suffit de choisir $R_1 := R$ et $R_2 := \emptyset$. □

Les applications des théorèmes 5.25 et 5.26 seront étudiées plus loin. Le théorème 5.26 se généralise de manière intéressante aux familles sans croisements :

Définition 5.27. *Soit G un graphe orienté et $\mathcal{F}$ une famille de sous-ensembles de $V(G)$. La* **matrice d'incidence des coupes sortantes** *de $\mathcal{F}$ est la matrice $M = (m_{X,e})_{X\in\mathcal{F},\, e\in E(G)}$ où*

$$m_{X,e} = \begin{cases} 1 & \textit{si } e \in \delta^+(X) \\ 0 & \textit{si } e \notin \delta^+(X) \end{cases}.$$

La **matrice d'incidence des coupes** *de $\mathcal{F}$ est la matrice $M = (m_{X,e})_{X\in\mathcal{F},\, e\in E(G)}$ où*

$$m_{X,e} = \begin{cases} -1 & \textit{si } e \in \delta^-(X) \\ 1 & \textit{si } e \in \delta^+(X) \\ 0 & \textit{sinon} \end{cases}.$$

Théorème 5.28. *Soit G un graphe orienté et $(V(G), \mathcal{F})$ une famille sans croisements d'ensembles. La matrice d'incidence des coupes de $\mathcal{F}$ est totalement unimodulaire. Si $\mathcal{F}$ est laminaire, la matrice d'incidence des coupes sortantes de $\mathcal{F}$ est également totalement unimodulaire.*

Preuve. Soit $\mathcal{F}$ une famille sans croisements de sous-ensembles de $V(G)$. Supposons d'abord que cette famille soit $\mathcal{F}$ laminaire.

Montrons que le critère du théorème 5.24 est satisfait. Soit $\mathcal{R} \subseteq \mathcal{F}$, et considérons la représentation par arbre (T, φ) de $\mathcal{R}$, où T est une arborescence de racine r (proposition 2.14). Utilisant les notations de la définition 2.13, $\mathcal{R} = \{S_e : e \in E(T)\}$. Soit $\mathcal{R}_1 := \{S_{(v,w)} \in \mathcal{R} : \mathrm{dist}_T(r,w) \text{ pair}\}$ et $\mathcal{R}_2 := \mathcal{R} \setminus \mathcal{R}_1$. Si $f \in E(G)$, les arcs $e \in E(T)$ tels que $f \in \delta^+(S_e)$ induisent un chemin P_f dans T (de longueur éventuelle zéro). Donc

$$|\{X \in \mathcal{R}_1 : f \in \delta^+(X)\}| - |\{X \in \mathcal{R}_2 : f \in \delta^+(X)\}| \in \{-1, 0, 1\},$$

ce qui montre la validité du critère du théorème 5.24 pour les coupes sortantes.

De plus, pour tout arc f les arcs $e \in E(T)$ avec $f \in \delta^-(S_e)$ induisent un chemin Q_f dans T. Puisque P_f et Q_f ont une extrémité commune,

$$\begin{aligned} &|\{X \in \mathcal{R}_1 : f \in \delta^+(X)\}| - |\{X \in \mathcal{R}_2 : f \in \delta^+(X)\}| \\ -&|\{X \in \mathcal{R}_1 : f \in \delta^-(X)\}| + |\{X \in \mathcal{R}_2 : f \in \delta^-(X)\}| \quad \in \quad \{-1, 0, 1\}, \end{aligned}$$

et le critère du théorème 5.24 est également satisfait pour les coupes.

Si $(V(G), \mathcal{F})$ est une famille sans croisements d'ensembles, soit

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{V(G) \setminus X : X \in \mathcal{F}, r \in X\}$$

pour un $r \in V(G)$ fixé. $\mathcal{F}'$ est laminaire. La matrice d'incidence des coupes de $\mathcal{F}$ est une sous-matrice de $\begin{pmatrix} M \\ -M \end{pmatrix}$, où M est la matrice d'incidence des coupes de $\mathcal{F}'$ et elle est donc également totalement unimodulaire. □

La matrice d'incidence des coupes sortantes n'est pas en général totalement unimodulaire (voir exercice 13). Pour une condition nécessaire et suffisante, voir Schrijver [1983]. La **matrice d'incidence des coupes** d'un graphe orienté est également appelée matrice de flot (exercice 14).

Seymour [1980] a montré que toutes les matrices totalement unimodulaires peuvent se construire à partir des matrices de flot et de deux autres matrices totalement unimodulaires. On peut vérifier en temps polynomial, grâce à ce résultat profond, si une matrice est totalement unimodulaire (voir Schrijver [1986]).

5.5 Plans coupants

Nous avons étudié les polyèdres entiers dans les paragraphes précédents. En général, si P est un polyèdre, nous avons $P \supset P_I$. Pour résoudre un programme en nombres entiers $\max\{cx : x \in P_I\}$, il est naturel d'essayer d'éliminer certaines parties de P afin d'obtenir de nouveau un polyèdre P' tel que $P \supset P' \supset P_I$. On peut espérer que $\max\{cx : x \in P'\}$ est atteint par un vecteur entier ; on pourra sinon obtenir un nouveau polyèdre à P'' à partir de P', en recommençant cette procédure d'élimination, et ainsi de suite. C'est l'idée principale de la méthode des

plans coupants, proposée initialement pour un problème particulier (le PROBLÈME DU VOYAGEUR DE COMMERCE) par Dantzig, Fulkerson et Johnson [1954].

Gomory [1958, 1963] a proposé un algorithme de résolution de la PROGRAMMATION EN NOMBRES ENTIERS grâce à la méthode des plans coupants. Dans ce paragraphe, nous n'étudierons que les aspects théoriques de cette méthode. L'algorithme de Gomory n'est pas polynomial et a peu d'applications pratiques en général. Cependant, l'idée originale de la méthode des plans coupants est souvent utilisée avec succès, en pratique. Cela sera étudié au paragraphe 21.6. La présentation suivante est principalement fondée sur Schrijver [1986].

Définition 5.29. *Si $P = \{x : Ax \leq b\}$ est un un polyèdre, soit*

$$P' := \bigcap_{P \subseteq H} H_I,$$

pour tous les demi-espaces rationnels affines $H = \{x : cx \leq \delta\}$ contenants P. Posons $P^{(0)} := P$ et $P^{(i+1)} := \left(P^{(i)}\right)'$. $P^{(i)}$ est appelée i-ième **troncature de Gomory-Chvátal** *de P.*

Si P est un polyèdre rationnel P, trivialement $P \supseteq P' \supseteq P^{(2)} \supseteq \cdots \supseteq P_I$ et $P_I = (P')_I$.

Proposition 5.30. *Si $P = \{x : Ax \leq b\}$ est un polyèdre rationnel,*

$$P' \;=\; \{x : uAx \leq \lfloor ub \rfloor \text{ quel que soit } u \geq 0 \text{ tel que } uA \text{ soit entier}\}.$$

Preuve. Faisons d'abord deux observations. Pour tout demi-espace affine rationnel $H = \{x : cx \leq \delta\}$ avec c entier, il est évident que

$$H' \;=\; H_I \;\subseteq\; \{x : cx \leq \lfloor \delta \rfloor\}. \tag{5.5}$$

Montrons que, si les composantes de c sont des nombres relativement premiers,

$$H' \;=\; H_I \;=\; \{x : cx \leq \lfloor \delta \rfloor\}. \tag{5.6}$$

Pour montrer (5.6), soit c un vecteur entier dont les composantes sont des nombres relativement premiers. Par le lemme 5.11, l'hyperplan $\{x : cx = \lfloor \delta \rfloor\}$ contient un vecteur entier y. Pour tout vecteur rationnel $x \in \{x : cx \leq \lfloor \delta \rfloor\}$, soit $\alpha \in \mathbb{N}$ tel que αx soit entier. On a alors

$$x \;=\; \frac{1}{\alpha}(\alpha x - (\alpha - 1)y) + \frac{\alpha - 1}{\alpha} y,$$

c.-à-d. x est combinaison convexe de points entiers qui sont dans H. Donc $x \in H_I$, ce qui prouve (5.6).

Démontrons maintenant le résultat. Pour vérifier l'implication $\subseteq$, observons que pour tout $u \geq 0$, $\{x : uAx \leq ub\}$ est un demi-espace contenant P, et par (5.5) $P' \subseteq \{x : uAx \leq \lfloor ub \rfloor\}$ si uA est entier.

Montrons maintenant l'implication $\supseteq$. Si $P = \emptyset$ le résultat est évident ; supposons donc $P \neq \emptyset$. Soit $H = \{x : cx \leq \delta\}$ un demi-espace affine rationnel contenant P. On peut supposer que c est entier et que les composantes de c sont des nombres relativement premiers. Notons que

$$\delta \;\geq\; \max\{cx : Ax \leq b\} \;=\; \min\{ub : uA = c, u \geq 0\}.$$

Soit u^* une solution optimale du problème de minimisation. Pour tout

$$z \;\in\; \{x : uAx \leq \lfloor ub \rfloor \text{ si } u \geq 0 \text{ et } uA \text{ est entier}\} \;\subseteq\; \{x : u^*Ax \leq \lfloor u^*b \rfloor\}$$

nous avons :

$$cz \;=\; u^*Az \;\leq\; \lfloor u^*b \rfloor \;\leq\; \lfloor \delta \rfloor,$$

ce qui, en utilisant (5.6), implique $z \in H_I$. □

Nous verrons plus loin que, pour tout polyèdre rationnel P, il existe un nombre t tel que $P_I = P^{(t)}$. La méthode des plans coupants de Gomory résout successivement les PLs sur P, P', P'', etc., jusqu'à obtention d'une solution optimale entière. À chaque étape, un nombre fini de nouvelles inégalités, définissant un système TDI pour le nouveau polyèdre, est ajouté (voir théorème 5.17) :

Théorème 5.31. (Schrijver [1980]) *Soit $P = \{x : Ax \leq b\}$ un polyèdre tel que $Ax \leq b$ est TDI, avec A entier et b rationnel. Alors $P' = \{x : Ax \leq \lfloor b \rfloor\}$. En particulier, si P est un polyèdre rationnel, P' est un polyèdre.*

Preuve. L'affirmation étant triviale si P est vide, supposons $P \neq \emptyset$. Clairement $P' \subseteq \{x : Ax \leq \lfloor b \rfloor\}$. Pour montrer l'autre inclusion, soit $u \geq 0$ un vecteur avec uA entier. Montrons que $uAx \leq \lfloor ub \rfloor$ pour tout x tel que $Ax \leq \lfloor b \rfloor$ (voir proposition 5.30) :

Nous savons que

$$ub \;\geq\; \max\{uAx : Ax \leq b\} \;=\; \min\{yb : y \geq 0,\; yA = uA\}.$$

Puisque $Ax \leq b$ est TDI, le minimum est atteint par un vecteur entier y^*. $Ax \leq \lfloor b \rfloor$ implique

$$uAx \;=\; y^*Ax \;\leq\; y^*\lfloor b \rfloor \;\leq\; \lfloor y^*b \rfloor \;\leq\; \lfloor ub \rfloor.$$

La seconde condition de l'énoncé se déduit du théorème 5.17. □

Pour obtenir le résultat principal de ce paragraphe, établissons les deux lemmes :

Lemme 5.32. *Si F est une face d'un polyèdre rationnel P, alors $F' = P' \cap F$. Plus généralement, $F^{(i)} = P^{(i)} \cap F$ pour tout $i \in \mathbb{N}$.*

Preuve. Soit $P = \{x : Ax \leq b\}$ avec A entier, b rationnel, et $Ax \leq b$ TDI (on se reportera au théorème 5.17).

Soit $F = \{x : Ax \leq b,\; ax = \beta\}$ une face de P, où $ax \leq \beta$ est une inégalité valide pour P avec a et β entiers.

Par la proposition 5.16, $Ax \leq b$, $ax \leq \beta$ est TDI, et donc par le théorème 5.18, $Ax \leq b$, $ax = \beta$ est aussi TDI. Comme β est entier,

$$\begin{aligned} P' \cap F &= \{x : Ax \leq \lfloor b \rfloor,\ ax = \beta\} \\ &= \{x : Ax \leq \lfloor b \rfloor,\ ax \leq \lfloor \beta \rfloor,\ ax \geq \lceil \beta \rceil\} \\ &= F'. \end{aligned}$$

Ici le théorème 5.31 a été utilisé deux fois.

Notons que soit F' est vide, soit F' est une face de P'. La fin de la preuve se fait alors par induction sur i : pour tout i soit $F^{(i)}$ est vide, soit $F^{(i)}$ est une face de $P^{(i)}$, et $F^{(i)} = P^{(i)} \cap F^{(i-1)} = P^{(i)} \cap (P^{(i-1)} \cap F) = P^{(i)} \cap F$. □

Lemme 5.33. *Soient P un polyèdre de $\mathbb{R}^n$ et U une matrice unimodulaire $n \times n$. Soit $f(P) := \{Ux : x \in P\}$. Alors $f(P)$ est un polyèdre. De plus, si P est un polyèdre rationnel, $(f(P))' = f(P')$ et $(f(P))_I = f(P_I)$.*

Preuve. Puisque $f : \mathbb{R}^n \to \mathbb{R}^n$, $x \mapsto Ux$ est une application linéaire bijective, la première partie du lemme est vraie. Puisque les restrictions de f et de f^{-1} à $\mathbb{Z}^n$ sont des bijections (par la proposition 5.9),

$$\begin{aligned} (f(P))_I &= \text{conv}(\{y \in \mathbb{Z}^n : y = Ux,\ x \in P\}) \\ &= \text{conv}(\{y \in \mathbb{R}^n : y = Ux,\ x \in P,\ x \in \mathbb{Z}^n\}) \\ &= \text{conv}(\{y \in \mathbb{R}^n : y = Ux,\ x \in P_I\}) \\ &= f(P_I). \end{aligned}$$

Soit $P = \{x : Ax \leq b\}$ tel que $Ax \leq b$ soit TDI, A étant une matrice entière et b étant rationnel (voir le théorème 5.17). Par définition $AU^{-1}x \leq b$ est aussi TDI. En utilisant le théorème 5.31 deux fois, il vient

$$(f(P))' = \{x : AU^{-1}x \leq b\}' = \{x : AU^{-1}x \leq \lfloor b \rfloor\} = f(P').$$

□

Théorème 5.34. (Schrijver [1980]) *Pour tout polyèdre rationnel P il existe un nombre t tel que $P^{(t)} = P_I$.*

Preuve. Soit P un polyèdre rationnel de $\mathbb{R}^n$. Montrons le théorème par induction sur $n + \dim P$. Le cas $P = \emptyset$ est évident et le cas $\dim P = 0$ est facile.

Supposons d'abord que P ne soit pas de pleine dimension. Alors $P \subseteq K$, K étant un hyperplan rationnel.

Si K ne contient aucun vecteur entier, il existe un vecteur entier a et un nombre non entier β tel que $K = \{x : ax = \beta\}$ (par le lemme 5.11). Mais alors $P' \subseteq \{x : ax \leq \lfloor \beta \rfloor,\ ax \geq \lceil \beta \rceil\} = \emptyset = P_I$.

Si K contient des vecteurs entiers : $K = \{x : ax = \beta\}$ avec a entier, et β entier, nous pouvons supposer que $\beta = 0$, parce que le théorème est invariant par translations par des vecteurs entiers. Par le lemme 5.10, il existe une matrice

unimodulaire U telle que $aU = \alpha e_1$. Puisque le théorème est aussi invariant par la transformation $x \mapsto U^{-1}x$ (par le lemme 5.33), nous pouvons supposer que $a = \alpha e_1$. Alors la première composante de tout vecteur de P est égale à zéro, et on peut donc réduire la dimension de l'espace d'une unité et utiliser l'hypothèse d'induction (notons que $(\{0\} \times Q)_I = \{0\} \times Q_I$ et $(\{0\} \times Q)^{(t)} = \{0\} \times Q^{(t)}$ pour tout polyèdre Q de $\mathbb{R}^{n-1}$ et tout $t \in \mathbb{N}$).

On peut donc supposer que $P = \{x : Ax \leq b\}$ est de pleine dimension et que A est entière. Par le théorème 5.1 il existe une matrice entière C et un vecteur d tel que $P_I = \{x : Cx \leq d\}$. Si $P_I = \emptyset$, posons $C := A$ et $d := b - A'\mathbb{1}$, ou A' s'obtient en remplaçant chaque coefficient de A par sa valeur absolue. (Notons que $\{x : Ax \leq b - A'\mathbb{1}\} = \emptyset$.)

Soit $cx \leq \delta$ une inégalité de $Cx \leq d$. Nous allons montrer qu'il existe $s \in \mathbb{N}$ tel que $P^{(s)} \subseteq H := \{x : cx \leq \delta\}$, ce qui montrera le théorème.

Observons d'abord qu'il existe $\beta \geq \delta$ tel que $P \subseteq \{x : cx \leq \beta\}$: si $P_I = \emptyset$, cela provient des choix de C et d ; si $P_I \neq \emptyset$, c'est une conséquence de la proposition 5.2.

Supposons, par contradiction, qu'il existe un entier γ avec $\delta < \gamma \leq \beta$ pour lequel il existe $s_0 \in \mathbb{N}$ avec $P^{(s_0)} \subseteq \{x : cx \leq \gamma\}$, mais pour lequel il n'existe aucun $s \in \mathbb{N}$ avec $P^{(s)} \subseteq \{x : cx \leq \gamma - 1\}$.

Observons que $\max\{cx : x \in P^{(s)}\} = \gamma$ pour tout $s \geq s_0$, car si $\max\{cx : x \in P^{(s)}\} < \gamma$ pour s, alors $P^{(s+1)} \subseteq \{x : cx \leq \gamma - 1\}$.

Soit $F := P^{(s_0)} \cap \{x : cx = \gamma\}$. F est une face de $P^{(s_0)}$, et $\dim F < n = \dim P$. Par l'hypothèse d'induction il existe un nombre s_1 tel que

$$F^{(s_1)} \;=\; F_I \;\subseteq\; P_I \cap \{x : cx = \gamma\} \;=\; \emptyset.$$

En appliquant le lemme 5.32 à F et $P^{(s_0)}$ nous obtenons

$$\emptyset \;=\; F^{(s_1)} \;=\; P^{(s_0+s_1)} \cap F \;=\; P^{(s_0+s_1)} \cap \{x : cx = \gamma\}.$$

Donc $\max\{cx : x \in P^{(s_0+s_1)}\} < \gamma$, et on obtient une contradiction. □

Ce théorème implique également le théorème suivant :

Théorème 5.35. (Chvátal [1973]) *Pour tout polytope P, il existe un nombre t tel que $P^{(t)} = P_I$.*

Preuve. Comme P est borné, il existe un polytope rationnel $Q \supseteq P$ tel que $Q_I = P_I$ (choisir un hypercube contenant P et prendre son intersection avec un demi-espace rationnel contenant P mais pas z, pour chaque point entier z appartenant à l'hypercube mais pas à P ; voir l'exercice 19 du chapitre 3). Par le théorème 5.34, il existe t tel que $Q^{(t)} = Q_I$. Donc $P_I \subseteq P^{(t)} \subseteq Q^{(t)} = Q_I = P_I$, ce qui implique $P^{(t)} = P_I$. □

Ce nombre t est appelé le rang de Chvátal de P. Si P n'est ni borné ni rationnel, on ne peut pas avoir de théorème de ce type : voir les exercices 1 et 17.

Un algorithme plus efficace, qui calcule l'enveloppe convexe entière d'un polyèdre de dimension deux a été trouvé par Harvey [1999]. Une version polynomiale de cette méthode des plans coupants, approchant une fonction objectif linéaire sur un polytope entier décrit par un oracle de séparation, a été proposée par Boyd [1997]. Cook, Kannan et Schrijver [1990] ont généralisé la procédure de Gomory-Chvátal à la programmation linéaire mixte. Citons aussi Eisenbrand [1999] qui a montré que vérifier si un vecteur rationnel donné appartient à P' pour un polyèdre rationnel donné P est un problème *coNP*-complet.

5.6 Relaxation lagrangienne

Supposons qu'un programme en nombres entiers $\max\{cx : Ax \leq b,\ A'x \leq b',$ x entier$\}$ soit beaucoup plus facile à résoudre quand les contraintes $A'x \leq b'$ ne sont pas prises en compte. Soit $Q := \{x \in \mathbb{Z}^n : Ax \leq b\}$ et supposons que l'on sache optimiser pour toute fonction objectif sur Q (par exemple si $\mathrm{conv}(Q) = \{x : Ax \leq b\}$). La relaxation lagrangienne est une technique qui permet de se débarrasser des contraintes gênantes (ici $A'x \leq b'$). Au lieu de tenir compte de ces contraintes, nous modifierons la fonction objectif afin de pénaliser les solutions non réalisables. Plus précisément, au lieu de résoudre

$$\max\{c^\top x : A'x \leq b',\ x \in Q\} \tag{5.7}$$

nous étudierons la fonction qui, à chaque vecteur $\lambda \geq 0$, associe

$$LR(\lambda) \ := \ \max\{c^\top x + \lambda^\top (b' - A'x) : x \in Q\}. \tag{5.8}$$

Pour chaque $\lambda \geq 0$, $LR(\lambda)$ est une borne supérieure pour (5.7) qui est relativement facile à calculer. (5.8) est appelé la **relaxation lagrangienne** de (5.7), et les composantes de λ sont appelés les **multiplicateurs de Lagrange**.

La relaxation lagrangienne est une technique utile en programmation non linéaire ; mais nous nous restreindrons ici à la programmation en nombres entiers.

On est bien entendu intéressé par l'obtention de la meilleure borne supérieure. Observons que $\lambda \mapsto LR(\lambda)$ est une fonction convexe. On peut utiliser la procédure suivante (appelée méthode des sous-gradients) pour résoudre $LR(\lambda)$:

Partons d'un vecteur arbitraire $\lambda^{(0)} \geq 0$. À l'itération i, $\lambda^{(i)}$ étant donné, trouver un vecteur $x^{(i)}$ qui maximise $c^\top x + (\lambda^{(i)})^\top (b' - A'x)$ sur Q (c.-à-d. calcule $LR(\lambda^{(i)})$). Notons que $LR(\lambda) - LR(\lambda^{(i)}) \geq (\lambda - \lambda^{(i)})^\top (b' - A'x^{(i)})$ pour tout λ, c.-à-d. que $b' - A'x^{(i)}$ est un sous-gradient de LR en $\lambda^{(i)}$. Posons $\lambda^{(i+1)} := \max\{0, \lambda^{(i)} - t_i(b' - A'x^{(i)})\}$ pour un certain $t_i > 0$. Polyak [1967] a montré que si $\lim_{i\to\infty} t_i = 0$ et $\sum_{i=0}^{\infty} t_i = \infty$, alors $\lim_{i\to\infty} LR(\lambda^{(i)}) = \min\{LR(\lambda) : \lambda \geq 0\}$. Pour d'autres résultats sur la convergence de la méthode des sous-gradients, on se réfèrera à Goffin [1977].

Le problème qui consiste à trouver la meilleure borne supérieure

$$\min\{LR(\lambda) : \lambda \geq 0\}$$

est quelquefois appelé la résolution du **dual lagrangien** de (5.7). Nous allons montrer que ce minimum est toujours atteint sauf si $\{x : Ax \leq b,\ A'x \leq b'\} = \emptyset$. Il faut aussi connaître la qualité de cette borne supérieure. Cela dépend de la structure du problème original. Nous rencontrerons au paragraphe 21.5 une application pour le PROBLÈME DU VOYAGEUR DE COMMERCE pour laquelle la relaxation lagrangienne est très efficace. Le théorème suivant nous aide à estimer la qualité de la borne supérieure :

Théorème 5.36. (Geoffrion [1974]) *Soit $c \in \mathbb{R}^n$, $A' \in \mathbb{R}^{m\times n}$ et $b' \in \mathbb{R}^m$. Soit $Q \subseteq \mathbb{R}^n$ tel que $\mathrm{conv}(Q)$ soit un polyèdre. Supposons que $\max\{c^\top x : A'x \leq b',\ x \in conv(Q)\}$ ait une solution optimale. Soit $LR(\lambda) := \max\{c^\top x + \lambda^\top(b' - A'x) : x \in Q\}$. Alors $\inf\{LR(\lambda) : \lambda \geq 0\}$ (la valeur optimale du dual lagrangien de $\max\{c^\top x : A'x \leq b',\ x \in Q\}$) est atteinte pour une valeur de λ, et est égale à $\max\{c^\top x : A'x \leq b',\ x \in \mathrm{conv}(Q)\}$.*

Preuve. Soit $\mathrm{conv}(Q) = \{x : Ax \leq b\}$. En utilisant le théorème de dualité en PROGRAMMATION LINÉAIRE 3.20 deux fois :

$$\begin{aligned}
& \max\{c^\top x : x \in \mathrm{conv}(Q),\ A'x \leq b'\} \\
= \ & \max\{c^\top x : Ax \leq b,\ A'x \leq b'\} \\
= \ & \min\{\lambda^\top b' + y^\top b : y^\top A + \lambda^\top A' = c^\top,\ y \geq 0,\ \lambda \geq 0\} \\
= \ & \min\{\lambda^\top b' + \min\{y^\top b : y^\top A = c^\top - \lambda^\top A',\ y \geq 0\} : \lambda \geq 0\} \\
= \ & \min\{\lambda^\top b' + \max\{(c^\top - \lambda^\top A')x : Ax \leq b\} : \lambda \geq 0\} \\
= \ & \min\{\max\{c^\top x + \lambda^\top(b' - A'x) : x \in \mathrm{conv}(Q)\} : \lambda \geq 0\} \\
= \ & \min\{\max\{c^\top x + \lambda^\top(b' - A'x) : x \in Q\} : \lambda \geq 0\} \\
= \ & \min\{LR(\lambda) : \lambda \geq 0\}.
\end{aligned}$$

La troisième ligne qui est un PL montre qu'il existe une valeur de λ pour laquelle le minimum est atteint. □

En particulier, si nous étudions un PL en nombres entiers $\max\{cx : A'x \leq b',\ Ax \leq b,\ x \text{ entier}\}$ tel que $\{x : Ax \leq b\}$ soit entier, alors le dual lagrangien (quand on relaxe $A'x \leq b'$ comme précédemment) conduit à la même borne supérieure que la relaxation standard $\max\{cx : A'x \leq b',\ Ax \leq b\}$. Si $\{x : Ax \leq b\}$ n'est pas entier, la borne supérieure est en général plus forte (mais peut être difficile à calculer) ; voir l'exercice 21 à titre d'exemple.

La relaxation lagrangienne peut aussi être utilisée pour approximer des PLs. Considérons par exemple le PROBLÈME D'AFFECTATION DES TÂCHES (voir (1.1), paragraphe 1.3). Le problème peut également s'écrire

$$\min\left\{T : \sum_{j\in S_i} x_{ij} \geq t_i\ (i = 1,\dots,n),\ (x,T) \in P\right\} \tag{5.9}$$

où P est le polytope

$$\left\{ (x,T) \;:\; 0 \le x_{ij} \le t_i \ (i=1,\dots,n,\ j \in S_i), \right.$$
$$\sum_{i:j\in S_i} x_{ij} \le T \ (j=1,\dots,m),$$
$$\left. T \le \sum_{i=1}^{n} t_i \right\}.$$

Appliquons maintenant la relaxation lagrangienne ; soit

$$LR(\lambda) \;:=\; \min \left\{ T + \sum_{i=1}^{n} \lambda_i \left(t_i - \sum_{j\in S_i} x_{ij} \right) : (x,T) \in P \right\}. \tag{5.10}$$

Ce PL à cause de sa structure particulière peut être résolu par un algorithme combinatoire simple (voir exercice 23), pour λ arbitraire. Si Q est l'ensemble des sommets de P (voir corollaire 3.32), en appliquant le théorème 5.36, nous pouvons conclure que la valeur optimale du dual lagrangien $\max\{LR(\lambda) : \lambda \ge 0\}$ est égale à la valeur du PL (5.9).

Exercices

1. Soit $P := \left\{(x,y) \in \mathbb{R}^2 : y \le \sqrt{2}x\right\}$. Montrer que P_I n'est pas un polyèdre. Trouver un polyèdre P tel que la fermeture de P_I ne soit pas un polyèdre.
2. Soit $P = \{x \in \mathbb{R}^{k+l} : Ax \le b\}$ un polyèdre rationnel. Montrer que $\mathrm{conv}(P \cap (\mathbb{Z}^k \times \mathbb{R}^l))$ est un polyèdre.
   Indication : généraliser la preuve du théorème 5.1.
   Note : cela est la base de la programmation linéaire mixte ; voir Schrijver [1986].
* 3. Montrer l'analogue entier du théorème de Carathéodory (exercice 14 du chapitre 3) : si $C = \{x \in \mathbb{Q}^n : Ax \le 0\}$ est un cône polyédral, si $\{a_1,\dots,a_t\}$ est une base de Hilbert C, alors tout point entier de C est combinaison entière non négative de $2n-1$ des vecteurs de cette base.
   Indication : on s'intéressera à une solution de base optimale du PL $\max\{y\mathbb{1} : yA = x,\ y \ge 0\}$ et on arrondira ses composantes.
   Note : le nombre $2n-1$ a été amélioré et porté à $2n-2$ Sebő [1990]. Il ne peut pas être amélioré au-dessous de $\lfloor \frac{7}{6}n \rfloor$ (Bruns *et al.* [1999]).
   (Cook, Fonlupt et Schrijver [1986])
4. Soit $C = \{x : Ax \ge 0\}$ un cône polyédral et soit b un vecteur tel que $bx > 0$ pour tout $x \in C\setminus\{0\}$. Montrer qu'il existe une base de Hilbert minimale unique qui génère C.
   (Schrijver [1981])

5. Soit A une matrice $m \times n$ entière, soient b et c deux vecteurs, et soit y une solution optimale de $\max\{cx : Ax \leq b,\ x \text{ entier}\}$. Trouver une solution optimale z de $\max\{cx : Ax \leq b\}$ telle que $||y - z||_\infty \leq n\Xi(A)$.
(Cook *et al.* [1986])

6. Montrer que toute matrice unimodulaire se déduit d'une matrice identité par transformation unimodulaire.
Indication : voir la preuve du lemme 5.10.

* 7. Montrer qu'il existe un algorithme polynomial qui, étant donné une matrice entière A et un vecteur entier b, trouve un vecteur entier x tel que $Ax = b$ ou conclut qu'un tel vecteur n'existe pas.
Indication : voir les preuves des lemmes 5.10 et 5.11.

8. Considérons les deux systèmes

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Ils définissent le même polyèdre. Montrer que le premier est TDI mais pas le second.

9. Soient $a \neq 0$ un vecteur entier et β un nombre rationnel. Montrer que l'inégalité $ax \leq \beta$ est TDI si et seulement si les composantes de a sont des nombres relativement premiers.

10. Soit $Ax \leq b$ un système TDI, soit $k \in \mathbb{N}$ et soit $\alpha > 0$ rationnel. Prouver que $\frac{1}{k}Ax \leq \alpha b$ est aussi TDI. Montrer que $\alpha Ax \leq \alpha b$ n'est pas toujours TDI.

11. Utiliser le théorème 5.25 pour montrer le théorème de König 10.2 (voir exercice 2 du chapitre 11) :
La taille maximum d'un couplage dans un graphe biparti est égale à la taille minimum d'une couverture par les sommets.

12. Montrer que $A = \left(\begin{smallmatrix} 1 & 1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$ n'est pas totalement unimodulaire, mais que $\{x : Ax = b\}$ est entier pour tous les vecteurs entiers b.
(Nemhauser et Wolsey [1988])

13. Soit G le graphe orienté $(\{1,2,3,4\}, \{(1,3),(2,4),(2,1),(4,1),(4,3)\})$, et soit $\mathcal{F} := \{\{1,2,4\},\{1,2\},\{2\},\{2,3,4\},\{4\}\}$. Montrer que $(V(G), \mathcal{F})$ est sans croisements, mais que la matrice d'incidence des coupes sortantes de $\mathcal{F}$ n'est pas totalement unimodulaire.

* 14. Soient G et T des graphes orientés avec $V(G) = V(T)$ et supposons que le graphe non orienté associé de T soit un arbre. Soit $P(v,w)$ la chaîne unique de T reliant v à w pour $v, w \in V(G)$. Soit $M = (m_{f,e})_{f \in E(T),\, e \in E(G)}$ la matrice définie par

$$m_{(x,y),(v,w)} := \begin{cases} 1 & \text{si } (x,y) \in E(P(v,w)) \text{ et } (x,y) \in E(P(v,y)) \\ -1 & \text{si } (x,y) \in E(P(v,w)) \text{ et } (x,y) \in E(P(v,x)) \\ 0 & \text{si } (x,y) \notin E(P(v,w)) \end{cases}.$$

De telles matrices sont appelées matrices graphiques. Montrer que les matrices graphiques sont les matrices d'incidence des coupes de systèmes d'ensembles sans croisements.

15. Une matrice d'intervalle est une matrice 0-1 telle que les corfficients +1 soient consécutifs sur chaque ligne. Montrer que les matrices d'intervalle sont totalement unimodulaires.
Note : Hochbaum et Levin [2006] ont montré comment résoudre efficacement des problèmes d'optimisation ayant de telles matrices.

16. Considérons le problème d'empilement d'intervalles suivant : étant donné une suite d'intervalles $[a_i, b_i]$, $i = 1, \ldots, n$, avec des poids $c_1, \ldots, c_n$ et un nombre $k \in \mathbb{N}$, trouver un sous-ensemble d'intervalles de poids maximum tel qu'aucun point ne soit contenu dans plus que dans k d'entre eux.
    (a) Donner une formulation par PL de ce problème (sans contraintes d'intégrité).
    (b) Étudier le cas $k = 1$. Quelle signification combinatoire a le dual ? Montrer comment résoudre le dual par un algorithme combinatoire simple.
    (c) Utiliser (b) pour obtenir un algorithme pour le problème d'empilements d'intervalles dans le cas $k = 1$ qui s'exécute en un temps $O(n \log n)$.
    (d) Trouver un algorithme simple en $O(n \log n)$ pour k général avec des poids unité.

    Note : voir aussi l'exercice 11 du chapitre 9.

17. Soit $P := \{(x, y) \in \mathbb{R}^2 : y = \sqrt{2}x,\ x \geq 0\}$ et soit $Q := \{(x, y) \in \mathbb{R}^2 : y = \sqrt{2}x\}$. Montrer que $P^{(t)} = P \neq P_I$ pour tout $t \in \mathbb{N}$ et $Q' = \mathbb{R}^2$.

18. Soit P l'enveloppe convexe de trois points $(0, 0)$, $(0, 1)$ et $(k, \frac{1}{2})$ dans $\mathbb{R}^2$, où $k \in \mathbb{N}$. Montrer que $P^{(2k-1)} \neq P_I$ mais que $P^{(2k)} = P_I$.

* 19. Soit $P \subseteq [0, 1]^n$ un polytope contenu dans hypercube unité avec $P_I = \emptyset$. Montrer que $P^{(n)} = \emptyset$.
Note : Eisenbrand et Schulz [2003] ont montré que $P^{(\lceil n^2(1+\log n)\rceil)} = P_I$ pour tout polytope $P \subseteq [0, 1]^n$.

20. Dans cet exercice, nous appliquerons la relaxation lagrangienne aux systèmes d'équations linéaires. Soient Q un ensemble fini de vecteurs de $\mathbb{R}^n$, $c \in \mathbb{R}^n$ et $A' \in \mathbb{R}^{m \times n}$ et $b' \in \mathbb{R}^m$. Montrer que

$$\begin{aligned} & \min\{\max\{c^\top x + \lambda^\top (b' - A'x) : x \in Q\} : \lambda \in \mathbb{R}^m\} \\ = \ & \max\{c^\top y : y \in \operatorname{conv}(Q),\ A'y = b'\}. \end{aligned}$$

21. Soit le problème de localisation suivant : nous avons n clients avec des demandes $d_1, \ldots, d_n$, et m services pouvant être ouverts ou non. À chaque service $i = 1, \ldots, m$ est associé un coût d'ouverture f_i, une capacité u_i et une distance c_{ij} d'éloignement de chaque client $j = 1, \ldots, n$. Le problème est de décider quels sont les services à ouvrir et comment affecter les services ouverts aux clients. La demande totale des clients affectés à un service ne doit pas excéder

sa capacité. L'objectif est de minimiser la somme des coûts d'ouverture des services plus la somme des distances de chaque client à son service. En termes de PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS le problème peut ainsi être formulé

$$\min\left\{\sum_{i,j} c_{ij}x_{ij} + \sum_i f_i y_i : \sum_j d_j x_{ij} \le u_i y_i,\ \sum_i x_{ij} = 1,\ x_{ij}, y_i \in \{0,1\}\right\}.$$

Appliquer la relaxation lagrangienne de deux manières, d'abord en relaxant $\sum_j d_j x_{ij} \le u_i y_i$ pour tout i, puis en relaxant $\sum_i x_{ij} = 1$ pour tout j. Quel dual lagrangien fournit une meilleure borne ?
Note : les deux relaxations lagrangiennes peuvent être utilisées : voir l'exercice 7 du chapitre 17.

∗ 22. Soit le PROBLÈME DE LOCALISATION SANS CAPACITÉS suivant : les nombres n, m, f_i et c_{ij} ($i = 1, \ldots, m$, $j = 1, \ldots, n$) étant donnés, le problème peut se formuler ainsi

$$\min\left\{\sum_{i,j} c_{ij}x_{ij} + \sum_i f_i y_i : \sum_i x_{ij} = 1,\ x_{ij} \le y_i,\ x_{ij}, y_i \in \{0,1\}\right\}.$$

Si $S \subseteq \{1, \ldots, n\}$ $c(S)$ sera le coût des services offerts aux clients de S, c.-à-d.

$$\min\left\{\sum_{i,j} c_{ij}x_{ij} + \sum_i f_i y_i : \sum_i x_{ij} = 1 \text{ pour } j \in S,\ x_{ij} \le y_i,\ x_{ij}, y_i \in \{0,1\}\right\}.$$

Le problème d'affectation des coûts consiste à savoir si le coût total $c(\{1, \ldots, n\})$ peut être réparti sur les clients de telle sorte qu'aucun sous-ensemble ne paie plus que $c(S)$. En d'autres termes : existe-t-il des nombres $p_1, \ldots, p_n$ tels que $\sum_{j=1}^n p_j = c(\{1, \ldots, n\})$ et $\sum_{j \in S} p_j \le c(S)$ pour tout $S \subseteq \{1, \ldots, n\}$? Montrer que cela est vrai si et seulement si $c(\{1, \ldots, n\})$ est égal à

$$\min\left\{\sum_{i,j} c_{ij}x_{ij} + \sum_i f_i y_i : \sum_i x_{ij} = 1,\ x_{ij} \le y_i,\ x_{ij}, y_i \ge 0\right\},$$

c.-à-d. quand les conditions d'intégrité sont redondantes.
Indication : appliquer la relaxation lagrangienne pour résoudre le PL précédent. Pour chaque ensemble de multiplicateurs de Lagrange, décomposer le problème de minimisation associé à des problèmes de minimisation sur des cônes polyédraux. Quels sont les vecteurs générant ces cônes ?
(Goemans et Skutella [2004])

23. Proposer un algorithme combinatoire (sans utiliser la PROGRAMMATION LINÉAIRE) pour résoudre (5.10) pour des multiplicateurs de Lagrange λ arbitraires mais fixés. Quel temps de calcul peut-on atteindre ?

Références

Littérature générale :

Bertsimas, D., Weismantel, R. [2005] : Optimization Over Integers. Dynamic Ideas, Belmont 2005

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. [1998] : Combinatorial Optimization. Wiley, New York 1998, Chapter 6

Nemhauser, G.L., Wolsey, L.A. [1988] : Integer and Combinatorial Optimization. Wiley, New York 1988

Schrijver, A. [1986] : Theory of Linear and Integer Programming. Wiley, Chichester 1986

Wolsey, L.A. [1998] : Integer Programming. Wiley, New York 1998

Références citées :

Boyd, E.A. [1997] : A fully polynomial epsilon approximation cutting plane algorithm for solving combinatorial linear programs containing a sufficiently large ball. Operations Research Letters 20 (1997), 59–63

Bruns, W., Gubeladze, J., Henk, M., Martin, A., Weismantel, R. [1999] : A counterexample to an integral analogue of Carathéodory's theorem. Journal für die Reine und Angewandte Mathematik 510 (1999), 179–185

Chvátal, V. [1973] : Edmonds' polytopes and a hierarchy of combinatorial problems. Discrete Mathematics 4 (1973), 305–337

Cook, W. [1983] : Operations that preserve total dual integrality. Operations Research Letters 2 (1983), 31–35

Cook, W., Fonlupt, J., Schrijver, A. [1986] : An integer analogue of Carathéodory's theorem. Journal of Combinatorial Theory B 40 (1986), 63–70

Cook, W., Gerards, A., Schrijver, A., Tardos, É. [1986] : Sensitivity theorems in integer linear programming. Mathematical Programming 34 (1986), 251–264

Cook, W., Kannan, R., Schrijver, A. [1990] : Chvátal closures for mixed integer programming problems. Mathematical Programming 47 (1990), 155–174

Dantzig, G., Fulkerson, R., Johnson, S. [1954] : Solution of a large-scale traveling-salesman problem. Operations Research 2 (1954), 393–410

Edmonds, J., Giles, R. [1977] : A min-max relation for submodular functions on graphs. In : Studies in Integer Programming ; Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, eds.), North-Holland, Amsterdam 1977, pp. 185–204

Eisenbrand, F. [1999] : On the membership problem for the elementary closure of a polyhedron. Combinatorica 19 (1999), 297–300

Eisenbrand, F., Schulz, A.S. [2003] : Bounds on the Chvátal rank of polytopes in the 0/1-cube. Combinatorica 23 (2003), 245–261

Fulkerson, D.R. [1971] : Blocking and anti-blocking pairs of polyhedra. Mathematical Programming 1 (1971), 168–194

Geoffrion, A.M. [1974] : Lagrangean relaxation for integer programming. Mathematical Programming Study 2 (1974), 82–114

Giles, F.R., Pulleyblank, W.R. [1979] : Total dual integrality and integer polyhedra. Linear Algebra and Its Applications 25 (1979), 191–196

Ghouila-Houri, A. [1962] : Caractérisation des matrices totalement unimodulaires. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris) 254 (1962), 1192–1194

Goemans, M.X., Skutella, M. [2004] : Cooperative facility location games. Journal of Algorithms 50 (2004), 194–214

Goffin, J.L. [1977] : On convergence rates of subgradient optimization methods. Mathematical Programming 13 (1977), 329–347

Gomory, R.E. [1958] : Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society 64 (1958), 275–278

Gomory, R.E. [1963] : An algorithm for integer solutions of linear programs. In : Recent Advances in Mathematical Programming (R.L. Graves, P. Wolfe, eds.), McGraw-Hill, New York, 1963, pp. 269–302

Graver, J.E. [1975] : On the foundations of linear and integer programming I. Mathematical Programming 9 (1975), 207–226

Harvey, W. [1999] : Computing two-dimensional integer hulls. SIAM Journal on Computing 28 (1999), 2285–2299

Hochbaum, D.S., Levin, A. [2006] : Optimizing over consecutive 1's and circular 1's constraints. SIAM Journal on Optimization 17 (2006), 311–330

Hoffman, A.J. [1974] : A generalization of max flow-min cut. Mathematical Programming 6 (1974), 352–359

Hoffman, A.J., Kruskal, J.B. [1956] : Integral boundary points of convex polyhedra. In : Linear Inequalities and Related Systems ; Annals of Mathematical Study 38 (H.W. Kuhn, A.W. Tucker, eds.), Princeton University Press, Princeton 1956, 223–246

Lasserre, J.B. [2004] : The integer hull of a convex rational polytope. Discrete & Computational Geometry 32 (2004), 129–139

Meyer, R.R. [1974] : On the existence of optimal solutions to integer and mixed-integer programming problems. Mathematical Programming 7 (1974), 223–235

Polyak, B.T. [1967] : A general method for solving extremal problems. Doklady Akademii Nauk SSSR 174 (1967), 33–36 [in Russian]. English translation : Soviet Mathematics Doklady 8 (1967), 593–597

Schrijver, A. [1980] : On cutting planes. In : Combinatorics 79 ; Part II ; Annals of Discrete Mathematics 9 (M. Deza, I.G. Rosenberg, eds.), North-Holland, Amsterdam 1980, pp. 291–296

Schrijver, A. [1981] : On total dual integrality. Linear Algebra and its Applications 38 (1981), 27–32

Schrijver, A. [1983] : Packing and covering of crossing families of cuts. Journal of Combinatorial Theory B 35 (1983), 104–128

Sebő, A. [1990] : Hilbert bases, Carathéodory's theorem and combinatorial optimization. In : Integer Programming and Combinatorial Optimization (R. Kannan and W.R. Pulleyblank, eds.), University of Waterloo Press, 1990

Seymour, P.D. [1980] : Decomposition of regular matroids. Journal of Combinatorial Theory B 28 (1980), 305–359

Veinott, A.F., Jr., Dantzig, G.B. [1968]. Integral extreme points. SIAM Review 10 (1968), 371–372

Wolsey, L.A. [1981] : The b-hull of an integer program. Discrete Applied Mathematics 3 (1981), 193–201

Chapitre 6

Arbres couvrants et arborescences

Un opérateur téléphonique souhaite louer un certain nombre de lignes parmi un ensemble de lignes existantes, chaque ligne reliant deux villes. Il s'agit de minimiser le coût de location sachant que toutes les villes doivent être connectées. On peut modéliser le réseau par un graphe ayant pour sommets les villes et pour arêtes, les lignes existantes. Par le théorème 2.4 les sous-graphes minimaux connexes engendrant un graphe donné sont les arbres couvrants de ce graphe. Nous cherchons donc un arbre couvrant de coût minimum, sachant que le coût d'un sous-graphe T d'un graphe G muni de coûts $c : E(G) \to \mathbb{R}$ est : $c(E(T)) = \sum_{e \in E(T)} c(e)$.

Ce problème simple est très important en optimisation combinatoire. C'est aussi un des problèmes ayant la plus longue histoire ; le premier algorithme a été proposé par Borůvka [1926a,1926b] ; voir Nešetřil, Milková et Nešetřilová [2001].

En comparaison avec le PROBLÈME DE PERÇAGE qui recherche une plus courte chaîne contenant tous les sommets d'un graphe complet, nous voulons trouver ici l'arbre couvrant le plus court. Bien que le nombre d'arbres couvrants soit supérieur au nombre de chaînes (K_n contient $\frac{n!}{2}$ chaînes hamiltoniennes, mais n^{n-2} arbres couvrants distincts par un théorème de Cayley [1889] ; voir exercice 1), il se trouve que ce problème est bien plus facile. Comme nous le verrons au paragraphe 6.1, une simple stratégie «gloutonne» résout le problème.

Les arborescences constituent la version orientée des arbres ; par le théorème 2.5, ce sont les sous-graphes couvrants minimaux d'un graphe orienté permettant d'atteindre tous les sommets à partir d'une racine. Le PROBLÈME DE L'ARBORESCENCE DE POIDS MINIMUM qui est la version orientée du PROBLÈME DE L'ARBRE COUVRANT MINIMUM est plus difficile à résoudre, la stratégie gloutonne n'étant plus adaptée. Nous montrerons comment résoudre ce problème au paragraphe 6.2.

Comme il existe de nombreux algorithmes combinatoires efficaces, nous n'utiliserons pas la PROGRAMMATION LINÉAIRE pour résoudre ces problèmes. Nous montrerons au paragraphe 6.3 qu'on sait décrire le polytope enveloppe convexe

des vecteurs d'incidence des arbres couvrants ou des arborescences (voir corollaire 3.33). Nous verrons au paragraphe 6.4 quelques résultats classiques concernant le problème de l'empilement d'arbres couvrants et d'arborescences.

6.1 Arbre couvrant de poids minimum

Dans ce paragraphe, nous étudierons les deux problèmes suivants :

PROBLÈME DE LA FORÊT DE POIDS MAXIMUM	
Instance	Un graphe non orienté G, un ensemble de poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver une forêt de G de poids maximum.

PROBLÈME DE L'ARBRE COUVRANT MINIMUM	
Instance	Un graphe non orienté G, un ensemble de poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver un arbre couvrant de G de poids minimum ou conclure que G n'est pas connexe.

Montrons d'abord que ces deux problèmes sont équivalents. Plus précisément, nous dirons qu'un problème $\mathcal{P}$ **se réduit linéairement** à un problème $\mathcal{Q}$ s'il existe des fonctions f et g calculables en temps linéaire telles que f transforme toute instance x de $\mathcal{P}$ en une instance $f(x)$ de $\mathcal{Q}$ et g transforme une solution de $f(x)$ en une solution de x. Si $\mathcal{P}$ se réduit linéairement à $\mathcal{Q}$ et $\mathcal{Q}$ se réduit linéairement à $\mathcal{P}$, ces deux problèmes seront appelés **équivalents**.

Proposition 6.1. *Le* PROBLÈME DE LA FORÊT DE POIDS MAXIMUM *et le* PROBLÈME DE L'ARBRE COUVRANT MINIMUM *sont équivalents.*

Preuve. Soit (G, c) une instance du PROBLÈME DE LA FORÊT DE POIDS MAXIMUM ; supprimons toutes les arêtes de poids négatif, posons $c'(e) := -c(e)$ pour toute arête de G de coût non négatif et ajoutons un ensemble minimum F d'arêtes (avec des poids arbitraires) afin que le graphe résultant G' soit connexe. L'instance (G', c') du PROBLÈME DE L'ARBRE COUVRANT MINIMUM est équivalente à l'instance (G, c) : en supprimant en effet F de l'arbre couvrant de poids minimum de (G', c'), nous obtenons une forêt de poids maximum de (G, c).

Inversement, soit (G, c) une instance du PROBLÈME DE L'ARBRE COUVRANT MINIMUM ; posons $c'(e) := K - c(e)$ pour $e \in E(G)$ ($K = 1 + \max_{e \in E(G)} c(e)$). L'instance (G, c') du PROBLÈME DE LA FORÊT DE POIDS MAXIMUM est équivalente à (G, c), puisque tous les arbres couvrants ont le même nombre d'arêtes. □

Nous reviendrons à d'autres exemples de réduction de problèmes au chapitre 15. Dans le reste de ce paragraphe nous étudierons uniquement le PROBLÈME DE L'ARBRE COUVRANT MINIMUM. Donnons d'abord deux conditions d'optimalité :

Théorème 6.2. *Si (G, c) est une instance du* PROBLÈME DE L'ARBRE COUVRANT MINIMUM*, et T est un arbre couvrant de G, les conditions suivantes sont équivalentes :*

(a) *T est optimum.*

(b) *Si $e = (x, y) \in E(G) \setminus E(T)$, aucune arête de la chaîne de x à y contenue dans T a un coût supérieur au coût de e.*

(c) *Si $e \in E(T)$, e a un coût minimum parmi les arêtes de l'ensemble $\delta(V(C))$, C étant une composante connexe de $T - e$.*

Preuve. (a)⇒(b) : si la condition (b) est violée, il existe $e = (x, y) \in E(G) \setminus E(T)$ et une arête f de la chaîne de x à y contenue dans T telle que $c(f) > c(e)$. Alors $(T - f) + e$ est un arbre couvrant de coût inférieur.

(b)⇒(c) : si la condition (c) est violée, il existe $e \in E(T)$, une composante connexe C de $T - e$ et $f = (x, y) \in \delta(V(C))$ tel que $c(f) < c(e)$. La chaîne de x à y contenue dans T contient une arête de $\delta(V(C))$ qui est forcément e. La condition (b) est donc violée.

(c)⇒(a) : supposons que T vérifie la condition (c), et soit T^* un arbre couvrant optimum tel que $E(T^*) \cap E(T)$ soit le plus grand possible. Montrons que $T = T^*$. Supposons qu'il existe $e = (x, y) \in E(T) \setminus E(T^*)$. Soit C une composante connexe de $T - e$. $T^* + e$ contient un cycle D. Puisque $e \in E(D) \cap \delta(V(C))$, il existe au moins une arête f ($f \neq e$) de D qui appartient à $\delta(V(C))$ (voir exercice 9 du chapitre 2). $(T^* + e) - f$ est un arbre couvrant. Puisque T^* est optimum, $c(e) \geq c(f)$. Mais puisque la condition (c) est vérifiée par T, $c(f) \geq c(e)$. Donc $c(f) = c(e)$, et $(T^* + e) - f$ est un autre arbre couvrant optimal ayant une arête de plus en commun avec T, ce qui est une contradiction. □

L'algorithme «glouton» suivant pour le PROBLÈME DE L'ARBRE COUVRANT MINIMUM, proposé par Kruskal [1956], peut être considéré comme un cas particulier de l'algorithme glouton général qui sera discuté au paragraphe 13.4. Dans la suite nous poserons $n := |V(G)|$ et $m := |E(G)|$.

ALGORITHME DE KRUSKAL	
Input	Un graphe connexe non orienté G, des coûts $c : E(G) \to \mathbb{R}$.
Output	Un arbre couvrant T de poids minimum.

① Trier les arêtes de telle sorte que $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$.

② Poser $T := (V(G), \emptyset)$.

③ **For** $i := 1$ **to** m **do** :
If $T + e_i$ ne contient pas de cycle **then** faire $T := T + e_i$.

Théorème 6.3. L'ALGORITHME DE KRUSKAL *répond correctement.*

Preuve. Il est clair que l'algorithme construit un arbre couvrant T et que la condition (b) du théorème 6.2 est vérifiée ; T est donc optimal. □

Le temps de calcul de l'ALGORITHME DE KRUSKAL est $O(mn)$: le tri des arêtes s'effectue en un temps $O(m \log m)$ (voir théorème 1.5), et l'algorithme pour trouver un cycle ayant au plus n arêtes dans un graphe peut s'implémenter en un temps $O(n)$ (il suffit d'appliquer les méthodes DFS (ou BFS) et vérifier s'il existe une arête qui n'appartient pas à l'ARBRE-DFS). Puisque cette opération se répète m fois, nous obtenons un temps total de calcul en $O(m \log m + mn) = O(mn)$. Cependant une meilleure implémentation est possible :

Théorème 6.4. *L'*ALGORITHME DE KRUSKAL *peut être exécuté en un temps* $O(m \log n)$.

Preuve. On peut éliminer les arêtes parallèles et ne garder qu'une arête de plus faible coût entre chaque paire de sommets adjacents. On peut donc supposer que $m = O(n^2)$. Puisque le temps de calcul de ① est $O(m \log m) = O(m \log n)$, concentrons nous sur ③. Implémentons une structure de données qui mémorise les composantes connexes de T. Dans ③, l'addition d'une arête $e_i = (v, w)$ à T crée un cycle si et seulement si v et w sont dans la même composante connexe.

Nous allons construire et mettre à jour une ramification B telle que $V(B) = V(G)$. À tout moment les composantes connexes de B seront induites par les ensembles des sommets des composantes connexes de T. (Il faut cependant remarquer que B n'est pas nécessairement une orientation de T.)

Quand on introduit une arête $e_i = (v, w)$ dans ③, on cherche la racine r_v de l'arborescence de B contenant v et la racine r_w de l'arborescence de B contenant w. Le temps pour cette opération est proportionnel à la longueur du chemin de r_v à v additionné de la longueur du chemin de r_w à w dans B. Nous montrerons plus tard que cette longueur est au plus $\log n$.

Vérifions ensuite si $r_v = r_w$. Si $r_v \neq r_w$, nous insérons e_i dans T et nous ajoutons un arc à B. Soit $h(r)$ la longueur maximum d'un chemin d'origine r dans B. Si $h(r_v) \geq h(r_w)$, nous ajoutons un arc (r_v, r_w) à B, sinon nous ajoutons un arc (r_w, r_v) à B. Si $h(r_v) = h(r_w)$, cette opération ajoute 1 à $h(r_v)$; sinon, la nouvelle racine garde la même h-valeur que précédemment. Ainsi les h-valeurs des racines sont facilement mises à jour. D'autre part la solution initiale est : $B := (V(G), \emptyset)$ et $h(v) := 0$ pour tout $v \in V(G)$.

Montrons qu'une arborescence de B ayant r comme racine contient au moins $2^{h(r)}$ sommets. Cela impliquera que $h(r) \leq \log n$ et prouvera le résultat. Cette propriété est vérifiée pour la solution initiale. Montrons qu'elle reste vraie quand on ajoute un arc (x, y) à B. Cela est trivial si $h(x)$ ne change pas. Sinon, $h(x) = h(y)$ avant l'opération d'addition, ce qui implique que chacune des deux arborescences contient au moins $2^{h(x)}$ sommets. La nouvelle arborescence de racine x contient au moins $2 \cdot 2^{h(x)} = 2^{h(x)+1}$ sommets, ce qui montre le résultat. □

L'implémentation précédente peut être encore améliorée : une fois connue la racine r_v de l'arborescence de B contenant v, tous les arcs situés sur le chemin P de r_v à v sont supprimés et un arc (r_x, x) est créé pour tout $x \in V(P) \setminus \{r_v\}$. Une analyse approfondie montre que cette heuristique de compression de chemin

rend le temps de calcul dans ③ presque linéaire : $O(m\alpha(m,n))$, où $\alpha(m,n)$ est la fonctionnelle inverse de la fonction d'Ackermann (voir Tarjan [1975,1983]).

Nous allons maintenant étudier un autre algorithme bien connu pour le PROBLÈME DE L'ARBRE COUVRANT MINIMUM, dû à Jarník [1930] (voir Korte et Nešetřil [2001]), Dijkstra [1959] et Prim [1957] :

ALGORITHME DE PRIM	
Input	Un graphe connexe non orienté G, des coûts $c : E(G) \to \mathbb{R}$.
Output	Un arbre couvrant T de coût minimum.

① Choisir $v \in V(G)$. Poser $T := (\{v\}, \emptyset)$.

② **While** $V(T) \neq V(G)$ **do** :
Choisir une arête $e \in \delta_G(V(T))$ de coût minimum. Poser $T := T + e$.

Théorème 6.5. *L'*ALGORITHME DE PRIM *répond correctement. Il s'exécute en un temps* $O(n^2)$.

Preuve. La condition (c) du théorème 6.2 est en effet satisfaite.

Afin d'obtenir un temps de calcul $O(n^2)$, pour chaque sommet $v \in V(G)\backslash V(T)$ nous ne conservons que l'arête $e \in E(V(T), \{v\})$ de moindre coût. Ces arêtes seront appelées candidats. Trouver initialement les candidats nécessite un temps $O(m)$. Chaque sélection de l'arête de moindre coût parmi les candidats se fait en un temps $O(n)$. La réactualisation des candidats se fait en balayant les arêtes incidentes au sommet ajouté à $V(T)$ et nécessite donc un temps $O(n)$. Puisque la boucle de l'instruction ② s'effectue $n-1$ fois, la borne $O(n^2)$ est démontrée. □

Le temps de calcul peut être amélioré par l'utilisation de structures de données efficaces. Soit $l_{T,v} := \min\{c(e) : e \in E(V(T), \{v\})\}$. Nous gérons l'ensemble $\{(v, l_{T,v}) : v \in V(G) \setminus V(T),\ l_{T,v} < \infty\}$ dans une structure appelée file prioritaire ou tas ; l est appelé la clé de l'élément (v,l). Trois opérations sont autorisées sur un tas : insertion d'un élément, recherche puis suppression d'un élément (v,l) ayant une clé l minimum, et enfin décroissance de la clé l de l'élément (v,l). On peut alors décrire de la manière suivante l'ALGORITHME DE PRIM :

① Choisir $v \in V(G)$. Poser $T := (\{v\}, \emptyset)$.
Soit $l_w := \infty$ pour tout $w \in V(G) \setminus \{v\}$.

② **While** $V(T) \neq V(G)$ **do** :
For $e = (v,w) \in E(\{v\}, V(G) \setminus V(T))$ **do** :
If $c(e) < l_w < \infty$ **then** faire $l_w := c(e)$ et DÉCROISSANCE(w, l_w).
If $l_w = \infty$ **then** faire $l_w := c(e)$ et INSÉRER(w, l_w).
$(v, l_v) :=$ SUPPRESSION.
Soit $e \in E(V(T), \{v\})$ tel que $c(e) = l_v$. Faire $T := T + e$.

Il y a de nombreuses manières d'implémenter un tas. Une méthode très efficace, appelée le tas de Fibonacci, a été proposée par Fredman et Tarjan [1987]. Notre présentation suit Schrijver [2003] :

Théorème 6.6. *Il est possible d'engendrer une structure de données pour représenter un ensemble fini (initialement vide), où à chaque élément u est associé un nombre réel $d(u)$, appelé clé, et effectuer toute suite de :*

- *p opérations d'*INSERTION *(addition d'un élément u de clé $d(u)$) ;*
- *n opérations de* SUPPRESSION *(recherche et suppression d'un élément u tel que $d(u)$ soit minimum) ;*
- *m opérations de* DÉCROISSANCE *(faire décroître la clé $d(u)$ d'un élément jusqu'à une valeur donnée)*

en un temps $O(m + p + n \log p)$.

Preuve. L'ensemble, noté U, est stocké dans un tas de Fibonacci, c.-à-d. une ramification (U, E) et une fonction $\varphi : U \to \{0, 1\}$ ayant les propriétés suivantes :

(i) Si $(u, v) \in E$ alors $d(u) \leq d(v)$. (Cela est l'ordre du tas.)

(ii) Pour tout $u \in U$ les enfants de u sont numérotés $1, \ldots, |\delta^+(u)|$ de telle sorte que, si v est le k-ième enfant, $|\delta^+(v)| + \varphi(v) \geq k - 1$.

(iii) Si u et v sont des racines distinctes ($\delta^-(u) = \delta^-(v) = \emptyset$), $|\delta^+(u)| \neq |\delta^+(v)|$.

La condition (ii) implique :

(iv) Si le degré sortant d'un sommet u est supérieur ou égal à k, alors au moins $\sqrt{2}^k$ sommets sont connectés à u.

Nous montrons (iv) par induction sur k, le cas $k = 0$ étant trivial. Soit u un sommet tel que $|\delta^+(u)| \geq k \geq 1$, et soit v un enfant de u tel que $|\delta^+(v)| \geq k - 2$ (v existe par (ii)). Nous appliquons l'hypothèse d'induction à v dans (U, E) et à u dans $(U, E \setminus \{(u, v)\})$ et concluons qu'au moins $\sqrt{2}^{k-2}$ et $\sqrt{2}^{k-1}$ sommets sont connectés à u. (iv) se démontre en observant que $\sqrt{2}^k \leq \sqrt{2}^{k-2} + \sqrt{2}^{k-1}$.

En particulier, (iv) implique que $|\delta^+(u)| \leq 2 \log |U|$ pour tout $u \in U$. Nous pouvons donc, par la propriété (iii), associer aux racines de (U, E) une fonction $b : \{0, 1, \ldots, \lfloor 2 \log |U| \rfloor\} \to U$ telle que $b(|\delta^+(u)|) = u$ pour chaque racine u.

De plus, nous garderons en mémoire la double liste chaînée des enfants (dans un ordre arbitraire), un pointeur vers le prédécesseur immédiat (parent), s'il existe, et le degré sortant de chaque sommet. Montrons maintenant comment les opérations INSERTION, SUPPRESSION et DÉCROISSANCE sont implémentées.

INSERTION$(v, d(v))$ est implémenté en posant $\varphi(v) := 0$ et en appliquant

PLANT(v) :

① Poser $r := b(|\delta^+(v)|)$.
if r est une racine telle que $r \neq v$ et $|\delta^+(r)| = |\delta^+(v)|$ **then** :
if $d(r) \leq d(v)$ **then** ajouter (r, v) à E et PLANT(r).
if $d(v) < d(r)$ **then** ajouter (v, r) à E et PLANT(v).
else poser $b(|\delta^+(v)|) := v$.

Comme (U, E) est toujours une ramification, la récursion se termine. Notons également que (i), (ii) et (iii) restent vraies.

SUPPRESSION est implémenté par balayage de $b(i)$, $i = 0, \ldots, \lfloor 2 \log |U| \rfloor$ pour trouver un élément u avec $d(u)$ minimum, puis par suppression de u, de ses arêtes incidentes et par application de PLANT(v) à chaque (ancien) enfant v de u.

DÉCROISSANCE$(v, (d(v))$ est un peu plus compliqué. Soit P le plus long chemin dans (U, E) se terminant en v tel que pour tout sommet interne u, $\varphi(u) = 1$. Posons $\varphi(u) := 1 - \varphi(u)$ pour tout $u \in V(P) \setminus \{v\}$, supprimons de E tous les arcs de P et appliquons PLANT(z) pour chaque arc (y, z) supprimé.

Pour vérifier que (ii) reste vraie, il suffit de considérer le parent du premier sommet x de P, s'il existe. Mais alors x n'est pas une racine, et la valeur de $\varphi(x)$ est modifiée de 0 à 1, compensant le successeur perdu de x.

Estimons finalement le temps de calcul. Comme φ augmente au plus m fois (au plus une fois dans chaque opération DÉCROISSANCE), φ diminue au plus m fois. Donc la somme des longueurs des chemins P dans toutes les opérations DÉCROISSANCE est au plus $m + m$. Au plus $2m + 2n \log p$ arcs sont donc supprimés au total (puisque chaque opération SUPPRESSION peut supprimer jusqu'à $2 \log p$ arcs). Donc au plus $2m + 2n \log p + p - 1$ arcs sont insérés au total. Cela montre que le temps global de calcul est $O(m + p + n \log p)$. □

Corollaire 6.7. L'ALGORITHME DE PRIM *implémenté avec le tas de Fibonacci résout le* PROBLÈME DE L'ARBRE COUVRANT MINIMUM *en un temps* $O(m + n \log n)$.

Preuve. Il y a au plus $n-1$ opérations INSERTION, $n-1$ opérations SUPPRESSION, et m opérations DÉCROISSANCE. □

Avec une implémentation plus sophistiquée, on peut améliorer le temps de calcul jusqu'à $O(m \log \beta(n, m))$, où $\beta(n, m) = \min \left\{ i : \log^{(i)} n \leq \frac{m}{n} \right\}$; voir Fredman et Tarjan [1987], Gabow, Galil et Spencer [1989], et Gabow *et al.* [1986]. L'algorithme déterministe le plus rapide est dû à Chazelle [2000] avec un temps de calcul de $O(m\alpha(m, n))$, où α est l'inverse de la fonction d'Ackermann.

Avec un autre modèle de calcul, Fredman et Willard [1994] ont obtenu une borne linéaire. Il existe un algorithme randomisé qui trouve un arbre couvrant minimum avec une espérance de temps de calcul linéaire (Karger, Klein et Tarjan [1995] ; un tel algorithme, qui trouve toujours une solution optimale, est appelé algorithme de Las Vegas). Cet algorithme utilise une procédure (déterministe) afin de tester si un arbre couvrant donné est optimal ; un algorithme linéaire pour ce problème a été trouvé par Dixon, Rauch et Tarjan [1992] ; voir aussi King [1995].

Le PROBLÈME DE L'ARBRE COUVRANT MINIMUM pour les graphes planaires se résout, de manière déterministe, en temps linéaire (Cheriton et Tarjan [1976]). Le PROBLÈME DE L'ARBRE COUVRANT MINIMUM dans le cas d'un ensemble de n points dans le plan peut être résolu en un temps $O(n \log n)$ (exercice 9). L'ALGORITHME DE PRIM peut être efficace pour de telles instances puisqu'on peut utiliser des structures adéquates pour trouver les plus proches voisins dans le plan.

6.2 Arborescence de poids minimum

Des généralisations naturelles du PROBLÈME DE LA FORÊT DE POIDS MAXIMUM et du PROBLÈME DE L'ARBRE COUVRANT MINIMUM se formulent comme suit dans le cas orienté :

PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM	
Instance	Un graphe orienté G, un ensemble de poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver une ramification de poids maximum dans G.

PROBLÈME DE L'ARBORESCENCE DE POIDS MINIMUM	
Instance	Un graphe orienté G, un ensemble de poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver une arborescence couvrante de poids minimum dans G ou décider qu'une telle arborescence n'existe pas.

Nous souhaiterons quelquefois spécifier la racine :

PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM	
Instance	Un graphe orienté G, un sommet $r \in V(G)$, des poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver une arborescence couvrante de poids minimum enracinée en r dans G ou décider qu'une telle arborescence n'existe pas.

Comme dans le cas non orienté, ces trois problèmes sont équivalents :

Proposition 6.8. *Le* PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM*, le* PROBLÈME DE L'ARBORESCENCE DE POIDS MINIMUM *et le* PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM *sont tous équivalents.*

Preuve. Soit (G, c) une instance du PROBLÈME DE L'ARBORESCENCE DE POIDS MINIMUM ; posons $c'(e) := K - c(e)$ pour tout $e \in E(G)$, avec $K = 1 + \sum_{e \in E(G)} |c(e)|$. Alors l'instance (G, c') du PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM est équivalente puisque, si B, B' sont deux ramifications telles que $|E(B)| > |E(B')|$, alors $c'(B) > c'(B')$ (et les ramifications avec $n - 1$ arcs sont exactement les arborescences couvrantes).

Soit (G, c) une instance du PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM et soit $G' := (V(G) \dot\cup \{r\}, E(G) \cup \{(r, v) : v \in V(G)\})$. Posons $c'(e) := -c(e)$ pour $e \in E(G)$ et $c(e) := 0$ pour $e \in E(G') \setminus E(G)$. Alors l'instance (G', r, c') du PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM est équivalente.

Finalement, soit une instance (G, r, c) du PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM et soit $G' := (V(G) \dot\cup \{s\}, E(G) \cup \{(s, r)\})$

et $c((s,r)) := 0$. Alors l'instance (G', c) du PROBLÈME DE L'ARBORESCENCE DE POIDS MINIMUM est équivalente. □

Dans le reste de ce paragraphe nous nous intéresserons seulement au PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM. Ce problème n'est pas aussi simple que le PROBLÈME DE LA FORÊT DE POIDS MAXIMUM. Par exemple, toute forêt maximale est maximum, mais les arcs gras de la figure 6.1 constituent une ramification maximale qui n'est pas maximum.

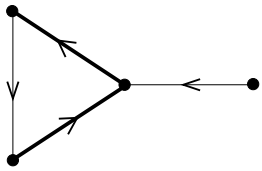


Figure 6.1.

Rappelons qu'une ramification est un graphe B tel que $|\delta_B^-(x)| \leq 1$ pour tout $x \in V(B)$ et tel que le graphe non orienté associé soit une forêt. De manière équivalente, une ramification est un graphe orienté B sans circuits tel que $|\delta_B^-(x)| \leq 1$ pour tout $x \in V(B)$ (voir théorème 2.5(g)) :

Proposition 6.9. *Soit B un graphe orienté tel que $|\delta_B^-(x)| \leq 1$ pour tout $x \in V(B)$. Alors B contient un circuit si et seulement si le graphe non orienté associé contient un cycle.* □

Soit G un graphe orienté et $c : E(G) \to \mathbb{R}_+$. Nous pouvons ignorer les arcs de coût négatif puisque aucun de ces arcs n'appartient à une ramification optimale. Une première idée d'algorithme est de considérer le meilleur arc entrant pour chaque sommet. Bien entendu, le graphe résultant peut contenir des circuits. Puisqu'une ramification ne contient pas de circuits, nous devons supprimer au moins un arc de chaque circuit. Le lemme suivant nous dit qu'on peut éliminer un seul arc.

Lemme 6.10. (Karp [1972]) *Soit B_0 un sous-graphe de G de poids maximum tel que $|\delta_{B_0}^-(v)| \leq 1$ pour tout $v \in V(B_0)$. Alors il existe une ramification optimale B de G telle que pour chaque circuit C de B_0, $|E(C) \setminus E(B)| = 1$.*

Preuve. Soit B une ramification optimale de G contenant le plus grand nombre possible d'arcs de B_0. Soit C un circuit de B_0.

Soit $E(C) \setminus E(B) = \{(a_1, b_1), \ldots, (a_k, b_k)\}$. Supposons $k \geq 2$; nous pouvons aussi supposer que $a_1, b_1, a_2, b_2, a_3, \ldots, b_k$ se situent dans cet ordre sur C (voir figure 6.2).

Montrons que B contient un chemin de b_i à b_{i-1} pour tout $i = 1, \ldots, k$ ($b_0 := b_k$), ce qui sera une contradiction puisque ces chemins induisent un parcours orienté fermé dans B, ce qui est impossible pour une ramification.

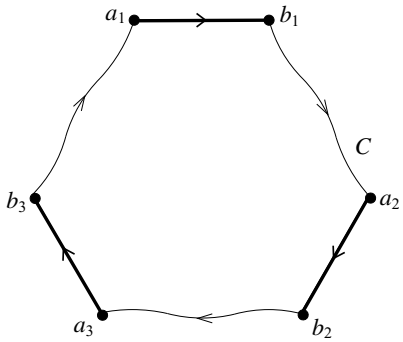


Figure 6.2.

Soit $i \in \{1, \ldots, k\}$. Il suffit de montrer que B contient un chemin de b_i à b_{i-1}. Considérons B' avec $V(B') = V(G)$ et $E(B') := \{(x, y) \in E(B) : y \neq b_i\} \cup \{(a_i, b_i)\}$.

B' ne peut être une ramification, sinon elle serait optimale et contiendrait plus d'arcs de B_0 que B. Donc, par la proposition 6.9 B' contient un circuit, c.-à-d. B contient un chemin P de b_i à a_i qui n'est pas contenu dans C, puisque $k \geq 2$; soit e le dernier arc de P qui n'appartient pas à C. Il est évident qu'il existe un sommet x tel que $e = (x, b_{i-1})$. Donc P (de même que B) contient un chemin de b_i à b_{i-1}. □

L'idée principale de l'algorithme d'Edmonds [1967] est de trouver d'abord B_0 et de contracter chaque circuit de B_0 dans G. Si nous choisissons correctement les poids dans le graphe résultant G_1, une ramification optimale dans G_1 correspondra à une ramification optimale dans G.

ALGORITHME DE RAMIFICATION D'EDMONDS	
Input	Un graphe orienté G, des poids $c : E(G) \to \mathbb{R}_+$.
Output	Une ramification de poids maximum B de G.

① Poser $i := 0$, $G_0 := G$, et $c_0 := c$.

② Soit B_i un sous-graphe de poids maximum G_i tel que $|\delta^-_{B_i}(v)| \leq 1$ pour tout $v \in V(B_i)$.

③ **If** B_i ne contient pas de circuits **then** poser $B := B_i$ et **go to** ⑤.

④ Construire (G_{i+1}, c_{i+1}) à partir de (G_i, c_i) en faisant ce qui suit pour chaque circuit C de B_i :
Contracter C en un seul sommet v_C dans G_{i+1}
For chaque arc $e = (z, y) \in E(G_i)$ avec $z \notin V(C)$, $y \in V(C)$ **do** :
Poser $z' = v_{C'}$ si z appartient à un circuit C' de B_i et $z' = z$ sinon.
Poser $e' := (z', v_C)$ et $\Phi(e') := e$.
$c_{i+1}(e') := c_i(e) - c_i(\alpha(e, C)) + c_i(e_C)$, où $\alpha(e, C)$
$= (x, y) \in E(C)$ et e_C est un arc de plus faible poids de C.
Faire $i := i + 1$ et **go to** ②.

⑤ **If** $i = 0$ **then stop**.

⑥ **For** chaque circuit C de B_{i-1} **do** :
If il existe un arc $e' = (z, v_C) \in E(B)$
then faire $E(B) := (E(B) \setminus \{e'\}) \cup \Phi(e') \cup (E(C) \setminus \{\alpha(\Phi(e'), C)\})$
else faire $E(B) := E(B) \cup (E(C) \setminus \{e_C\})$.
Faire $V(B) := V(G_{i-1})$, $i := i - 1$ et **go to** ⑤.

Cet algorithme a été découvert indépendamment par Chu et Liu [1965] et Bock [1971].

Théorème 6.11. (Edmonds [1967]) L'ALGORITHME DE RAMIFICATION D'EDMONDS *répond correctement.*

Preuve. Nous allons montrer qu'à chaque instant précédant l'exécution de ⑤, B est une ramification optimale de G_i. Cela est évident quand nous atteignons ⑤ pour la première fois. Nous avons donc à montrer que ⑥ transforme une ramification optimale B de G_i en une ramification optimale B' de G_{i-1}.

Soit B^*_{i-1} une ramification de G_{i-1} telle que $|E(C) \setminus E(B^*_{i-1})| = 1$ pour tout circuit C de B_{i-1}. Si B^*_i provient de B^*_{i-1} par contraction des circuits de B_{i-1}, B^*_i est une ramification de G_i. De plus

$$c_{i-1}(B^*_{i-1}) \;=\; c_i(B^*_i) + \sum_{C\text{: circuit de } B_{i-1}} (c_{i-1}(C) - c_{i-1}(e_C)).$$

Par l'hypothèse d'induction, B est une ramification optimale de G_i et donc $c_i(B) \geq c_i(B^*_i)$. Nous en déduisons que

$$\begin{aligned} c_{i-1}(B^*_{i-1}) \;&\leq\; c_i(B) + \sum_{C\text{: circuit de } B_{i-1}} (c_{i-1}(C) - c_{i-1}(e_C)) \\ &=\; c_{i-1}(B'). \end{aligned}$$

Cela, avec le lemme 6.10, implique que B' est une ramification optimale deG_{i-1}. □

Cette preuve est due à Karp [1972]. La preuve originale d'Edmonds était fondée sur une formulation par programmation linéaire (voir corollaire 6.14). Le temps de

calcul de L'ALGORITHME DE RAMIFICATION D'EDMONDS est $O(mn)$, où $m = |E(G)|$ et $n = |V(G)|$: en effet il y a au plus n itérations (c.-à-d. $i \leq n$ à chaque étape de l'algorithme), et chaque itération nécessite un temps $O(m)$.

La meilleure borne connue a été obtenue par Gabow *et al.* [1986] qui utilisent un tas de Fibonacci : leur algorithme de ramification s'exécute en $O(m + n \log n)$.

6.3 Descriptions polyédrales

Nous pouvons associer au PROBLÈME DE L'ARBRE COUVRANT MINIMUM la description polyédrale suivante :

Théorème 6.12. (Edmonds [1970]) *Étant donné un graphe non orienté connexe G, $n := |V(G)|$, le polytope $P :=$*

$$\left\{ x \in [0,1]^{E(G)} : \sum_{e \in E(G)} x_e = n-1, \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ pour } \emptyset \neq X \subset V(G) \right\}$$

est entier. Ses sommets sont les vecteurs d'incidence des arbres couvrants de G. (P est appelé le **polytope des arbres couvrants** *de G.)*

Preuve. Soit T un arbre couvrant de G, et soit x le vecteur d'incidence de $E(T)$. Il est évident (par le théorème 2.4) que $x \in P$. De plus, $x \in \{0,1\}^{E(G)}$; c'est donc un sommet de P.

Soit x un sommet entier de P ; x est le vecteur d'incidence d'un ensemble d'arêtes d'un sous-graphe H ayant $n-1$ arêtes et aucun cycle. De nouveau par le théorème 2.4, H est un arbre couvrant.

Il suffit donc de montrer que P est entier (voir théorème 5.13). Soit $c : E(G) \to \mathbb{R}$, et soit T l'arbre obtenu par l'ALGORITHME DE KRUSKAL appliqué à l'input (G, c). Soit $E(T) = \{f_1, \ldots, f_{n-1}\}$, où les f_i sont examinées dans cet ordre par l'algorithme. En particulier, $c(f_1) \leq \cdots \leq c(f_{n-1})$. Soit $X_k \subseteq V(G)$ la composante connexe de $(V(G), \{f_1, \ldots, f_k\})$ contenant f_k ($k = 1, \ldots, n-1$).

Soit x^* le vecteur d'incidence de $E(T)$. Montrons que x^* est une solution optimale du PL

$$\begin{array}{lll} \min & \displaystyle\sum_{e \in E(G)} c(e) x_e & \\ \text{s.c.} & \displaystyle\sum_{e \in E(G)} x_e = n-1 & \\ & \displaystyle\sum_{e \in E(G[X])} x_e \leq |X| - 1 & (\emptyset \neq X \subset V(G)) \\ & x_e \geq 0 & (e \in E(G)). \end{array}$$

Associons une variable duale z_X à chaque $\emptyset \neq X \subset V(G)$ et une variable duale additionnelle $z_{V(G)}$ à la contrainte d'égalité. Le PL dual est

$$\begin{array}{lll} \max & -\displaystyle\sum_{\emptyset \neq X \subseteq V(G)} (|X|-1) z_X & \\ \text{s.c.} & -\displaystyle\sum_{e \subseteq X \subseteq V(G)} z_X \leq c(e) & (e \in E(G)) \\ & z_X \geq 0 & (\emptyset \neq X \subset V(G)). \end{array}$$

Notons que la variable duale $z_{V(G)}$ n'est pas astreinte à être non négative. Pour $k = 1, \ldots, n-2$ soit $z^*_{X_k} := c(f_l) - c(f_k)$, où l est le premier indice k tel que $f_l \cap X_k \neq \emptyset$. Soit $z^*_{V(G)} := -c(f_{n-1})$, et posons $z^*_X := 0$ pour tout $X \notin \{X_1, \ldots, X_{n-1}\}$.

Pour chaque $e = (v, w)$,

$$-\sum_{e \subseteq X \subseteq V(G)} z^*_X = c(f_i),$$

i étant le plus petit indice tel que $v, w \in X_i$. De plus $c(f_i) \leq c(e)$ puisque v et w sont dans des composantes connexes distinctes de $(V(G), \{f_1, \ldots, f_{i-1}\})$. Donc z^* est une solution duale réalisable.

Si $e \in E(T)$, $x^*_e > 0$ et

$$-\sum_{e \subseteq X \subseteq V(G)} z^*_X = c(e),$$

c.-à-d. la contrainte duale correspondante est satisfaite avec égalité. Finalement, $z^*_X > 0$ implique que $T[X]$ est connexe, et la contrainte correspondante du primal est satisfaite avec égalité. Les conditions d'écarts complémentaires sont donc satisfaites et, par le corollaire 3.23, x^* et z^* sont respectivement solutions optimales des PL primal et dual. □

Notons que nous venons de montrer que le système d'inégalités dans le théorème 6.12 est TDI. Remarquons aussi que la démonstration précédente est une nouvelle preuve de l'exactitude de l'ALGORITHME DE KRUSKAL (théorème 6.3). Une autre description du polytope des arbres couvrants est l'objet de l'exercice 14.

En remplaçant la contrainte $\sum_{e \in E(G)} x_e = n-1$ par $\sum_{e \in E(G)} x_e \leq n-1$, nous obtenons l'enveloppe convexe des vecteurs d'incidence des forêts de G (exercice 15). Une généralisation de ces résultats est la caractérisation donnée par Edmonds du polytope des matroïdes (théorème 13.21).

Nous allons maintenant nous intéresser à une description polyédrale associée au PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM. Démontrons d'abord un résultat classique de Fulkerson. Rappelons qu'une coupe issue de r est un ensemble d'arcs $\delta^+(S)$ tel que $S \subset V(G)$ et $r \in S$.

Théorème 6.13. (Fulkerson [1974]) *Soient G un graphe orienté muni de poids $c : E(G) \to \mathbb{Z}_+$, et $r \in V(G)$ tel que G contienne une arborescence couvrante enracinée en r. Le poids minimum d'une arborescence couvrante enracinée en r est égal au nombre maximum t de coupes issues de r, $C_1, \ldots, C_t$ (les répétitions sont permises) tel qu'aucun arc e ne soit contenu dans plus que $c(e)$ de ces coupes.*

Preuve. Soit A la matrice dont les colonnes sont indicées par les arcs et dont les lignes sont les vecteurs d'incidence des coupes issues de r. Soit le PL

$$\min\{cx : Ax \geq \mathbb{1},\ x \geq 0\},$$

et son dual

$$\max\{\mathbb{1}y : yA \leq c, y \geq 0\}.$$

Nous devons alors montrer par la partie (e) du théorème 2.5 que, pour tout vecteur entier non négatif c, le primal et le dual ont des solutions optimales entières. Par le corollaire 5.15, il suffit de montrer que le système $Ax \geq \mathbb{1},\ x \geq 0$ est TDI. Nous utiliserons le lemme 5.23.

Puisque le PL dual est réalisable si et seulement si c est non négatif, soit $c : E(G) \to \mathbb{Z}_+$. Soit y une solution optimale de $\max\{\mathbb{1}y : yA \leq c, y \geq 0\}$ telle que

$$\sum_{\emptyset \neq X \subseteq V(G)\setminus\{r\}} y_{\delta^-(X)}|X|^2 \tag{6.1}$$

est aussi grand que possible. Montrons que $\mathcal{F} := \{X : y_{\delta^-(X)} > 0\}$ est laminaire. Supposons que $X, Y \in \mathcal{F}$ avec $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$ et $Y \setminus X \neq \emptyset$ (figure 6.3). Soit $\epsilon := \min\{y_{\delta^-(X)}, y_{\delta^-(Y)}\}$. Posons $y'_{\delta^-(X)} := y_{\delta^-(X)} - \epsilon$, $y'_{\delta^-(Y)} := y_{\delta^-(Y)} - \epsilon$, $y'_{\delta^-(X\cap Y)} := y_{\delta^-(X\cap Y)} + \epsilon$, $y'_{\delta^-(X\cup Y)} := y_{\delta^-(X\cup Y)} + \epsilon$, et $y'(S) := y(S)$ pour toutes les autres coupes issues de r et induites par S. Observons que $y'A \leq yA$ et que y' est une solution duale réalisable. Puisque $\mathbb{1}y = \mathbb{1}y'$, y' est aussi optimale, ce qui contredit notre choix de y, car (6.1) est plus grand pour y'. (Pour tous nombres $a > b \geq c > d > 0$ tels que $a + d = b + c$, $a^2 + d^2 > b^2 + c^2$.)

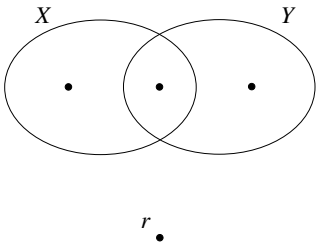


Figure 6.3.

Soit A' la sous-matrice de A dont les lignes sont indicées par les éléments de $\mathcal{F}$. A' est la matrice des coupes orientées d'une famille laminaire (plus précisément, il faut considérer le graphe déduit de G en inversant l'orientation des arcs). Par le théorème 5.28, A' est totalement unimodulaire, ce qui démontre le résultat. □

La preuve précédente fournit la description polyédrale suivante :

Corollaire 6.14. (Edmonds [1967]) *Soit G un graphe orienté muni de poids $c : E(G) \to \mathbb{R}_+$, et soit $r \in V(G)$ tel que G contienne une arborescence couvrante enracinée en r. Alors le PL*

$$\min\left\{cx : x \geq 0,\ \sum_{e\in\delta^+(X)} x_e \geq 1 \text{ pour tout } X \subset V(G) \text{ avec } r \in X\right\}$$

a une solution optimale entière, vecteur d'incidence d'une arborescence enracinée en r de poids minimum, avec l'addition éventuelle d'arcs de poids nul. □

Pour une description de l'enveloppe convexe des vecteurs d'incidence des ramifications ou des arborescences enracinées en r, voir les exercices 16 et 17.

6.4 Empilements d'arbres et d'arborescences

Si on s'intéresse à plusieurs arbres couvrants ou arborescences, les théorèmes classiques de Tutte, Nash-Williams et Edmonds sont utiles. Nous présentons d'abord une preuve du théorème de Tutte sur les empilements d'arbres couvrants, due principalement à Mader (voir Diestel [1997]), et qui utilise le lemme suivant :

Lemme 6.15. *Soit G un graphe non orienté, et soit $F = (F_1, \ldots, F_k)$ un k-uplet de forêts arête-disjointes dans G tel que $|E(F)|$ soit maximum, où $E(F) := \bigcup_{i=1}^{k} E(F_i)$. Soit $e \in E(G) \setminus E(F)$. Alors il existe un ensemble $X \subseteq V(G)$ avec $e \subseteq X$ tel que $F_i[X]$ soit connexe pour tout $i \in \{1, \ldots, k\}$.*

Preuve. Si $F' = (F'_1, \ldots, F'_k)$ et $F'' = (F''_1, \ldots, F''_k)$ sont deux k-uplets, nous dirons que F'' provient de F' par échange de e' avec e'' si $F''_j = (F'_j \setminus e') \cup e''$ pour un indice j et $F''_i = F'_i$ pour $i \neq j$. Soit $\mathcal{F}$ l'ensemble de tous les k-uplets de forêts arête-disjointes provenant de F par une succession de tels échanges. Soit $\overline{E} := E(G) \setminus \left(\bigcap_{F'\in\mathcal{F}} E(F')\right)$ et $\overline{G} := (V(G), \overline{E})$. $F \in \mathcal{F}$ et par conséquent $e \in \overline{E}$. Soit X l'ensemble des sommets de la composante connexe de $\overline{G}$ contenant e. Montrons que $F_i[X]$ est connexe pour chaque i.

Montrons d'abord que, pour tout $F' = (F'_1, \ldots, F'_k) \in \mathcal{F}$ et tout $\bar{e} = (v, w) \in E(\overline{G}[X]) \setminus E(F')$ il existe une chaîne de v à w dans $F'_i[X]$ pour tout $i \in \{1, \ldots, k\}$.

Pour prouver cela, soit $i \in \{1, \ldots, k\}$ fixé. Puisque $F' \in \mathcal{F}$ et $|E(F')| = |E(F)|$ est maximum, $F'_i + \bar{e}$ contient un cycle C. Pour tout $e' \in E(C) \setminus \{\bar{e}\}$, $F'_{e'} \in \mathcal{F}$, où $F'_{e'}$ provient de F' par échange de e' avec $\bar{e}$. Donc $E(C) \subseteq \overline{E}$, et $C - \bar{e}$ est une chaîne de v à w dans $F'_i[X]$, ce qui montre le résultat.

Puisque $\overline{G}[X]$ est connexe, il suffit de montrer que, pour chaque $\bar{e} = (v, w) \in E(\overline{G}[X])$ et chaque i, il existe une chaîne de v à w dans $F_i[X]$.

Soit donc $\bar{e} = (v, w) \in E(\overline{G}[X])$. Puisque $\bar{e} \in \overline{E}$, il existe $F' = (F'_1, \ldots, F'_k) \in \mathcal{F}$ avec $\bar{e} \notin E(F')$. Par le résultat précédent, il existe une chaîne de v à w dans $F'_i[X]$ pour chaque i.

Il existe alors une suite $F = F^{(0)}, F^{(1)} \ldots, F^{(s)} = F'$ d'éléments de $\mathcal{F}$ telle que $F^{(r+1)}$ provienne de $F^{(r)}$ par échange d'une arête ($r = 0, \ldots, s-1$). Il suffit de montrer que l'existence d'une chaîne de v à w dans $F_i^{(r+1)}[X]$ implique l'existence d'une chaîne de v à w dans $F_i^{(r)}[X]$ ($r = 0, \ldots, s-1$).

Supposons donc $F_i^{(r+1)}[X]$ provienne de $F_i^{(r)}[X]$ par échange de e_r avec e_{r+1}, et soit P la chaîne de v à w dans $F_i^{(r+1)}[X]$. Si P ne contient pas $e_{r+1} = (x, y)$, P est une chaîne dans $F_i^{(r)}[X]$. Sinon $e_{r+1} \in E(\overline{G}[X])$; soit alors la chaîne Q de x à y dans $F_i^{(r)}[X]$ qui existe par le résultat précédent. Puisque $(E(P) \setminus \{e_{r+1}\}) \cup Q$ contient une chaîne de v à w dans $F_i^{(r)}[X]$, le résultat est démontré. □

Nous pouvons montrer maintenant le théorème de Tutte sur les arbres couvrants disjoints. Une **multicoupe** dans un graphe non orienté G est un ensemble d'arêtes $\delta(X_1, \ldots, X_p) := \delta(X_1) \cup \cdots \cup \delta(X_p)$ associé à une partition $V(G) = X_1 \dot\cup X_2 \dot\cup \cdots \dot\cup X_p$ de l'ensemble des sommets en sous-ensembles non vides. Si $p = 3$, nous parlerons de 3-coupes. Les coupes sont des multicoupes avec $p = 2$.

Théorème 6.16. (Tutte [1961], Nash-Williams [1961]) *Un graphe non orienté G contient k arbres couvrants arête-disjoints si et seulement si*

$$|\delta(X_1, \ldots, X_p)| \ \geq \ k(p-1)$$

pour toute multicoupe $\delta(X_1, \ldots, X_p)$.

Preuve. Pour montrer la nécessité, soient $T_1, \ldots, T_k$ des arbres couvrants arête-disjoints de G, et soit $\delta(X_1, \ldots, X_p)$ une multicoupe. En contractant les sous-ensembles de sommets $X_1, \ldots, X_p$, on obtient un graphe G' ayant pour sommets $X_1, \ldots, X_p$ et pour arêtes les arêtes de la multicoupe. À $T_1, \ldots, T_k$ correspondent des sous-graphes connexes arête-disjoints $T_1', \ldots, T_k'$ de G'. Chacun des $T_1', \ldots, T_k'$ a au moins $p-1$ arêtes, et G' ainsi que la multicoupe ont au moins $k(p-1)$ arêtes.

Pour montrer que la condition est suffisante, faisons une induction sur $|V(G)|$. Le résultat est vrai si $n := |V(G)| \leq 2$. Si $n > 2$, supposons que $|\delta(X_1, \ldots, X_p)| \geq k(p-1)$ pour toute multicoupe $\delta(X_1, \ldots, X_p)$. En particulier (en considérant la partition en singletons) G a au moins $k(n-1)$ arêtes. De plus, la condition reste vérifiée par contraction des sommets ; par l'hypothèse d'induction G/X contient k arbres couvrants arête-disjoints pour chaque $X \subset V(G)$ tel que $|X| \geq 2$.

Soit $F = (F_1, \ldots, F_k)$ un k-uplet de forêts arête-disjointes de G tel que $|E(F)|$ soit maximum, où $E(F) := \bigcup_{i=1}^k E(F_i)$. Si une des forêts F_i n'est pas un arbre couvrant, $|E(F)| < k(n-1)$, et il existe une arête $e \in E(G) \setminus E(F)$. Par le lemme 6.15, il existe $X \subseteq V(G)$ avec $e \subseteq X$ tel que $F_i[X]$ soit connexe pour chaque i. Puisque $|X| \geq 2$, G/X contient k arbres couvrants arête-disjoints $F_1', \ldots, F_k'$. L'union de F_i' et de $F_i[X]$ est un arbre couvrant de G pour tout i, et tous ces k arbres couvrants sont arête-disjoints, ce qui est une contradiction. □

Étudions maintenant le problème de l'empilement d'arborescences couvrantes :

Théorème 6.17. (Edmonds [1973]) *Soit G un graphe orienté et soit $r \in V(G)$. Le nombre maximum d'arborescences arc-disjointes enracinées en r est égal à la cardinalité minimum d'une coupe issue de r.*

Preuve. Soit k la cardinalité minimum d'une coupe issue de r. Il est évident qu'il existe au plus k arborescences couvrantes arc-disjointes. Montrons par induction sur k qu'il en existe exactement k. Le cas $k = 0$ est trivial.

S'il existe une arborescence couvrante A enracinée en r telle que

$$\min_{r \in S \subset V(G)} \left| \delta_G^+(S) \setminus E(A) \right| \ \geq \ k - 1, \tag{6.2}$$

alors la preuve sera terminée par induction. Supposons que nous ayons trouvé une arborescence A enracinée en r (mais non nécessairement couvrante) telle que (6.2) soit vérifié. Soit $R \subseteq V(G)$ l'ensemble des sommets couverts par A. Initialement, $R = \{r\}$; si $R = V(G)$, la preuve est terminée.

Si $R \neq V(G)$, nous dirons qu'un ensemble $X \subseteq V(G)$ est critique si :

(a) $r \in X$.

(b) $X \cup R \neq V(G)$.

(c) $|\delta_G^+(X) \setminus E(A)| = k - 1$.

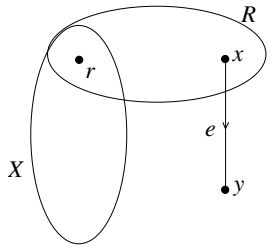


Figure 6.4.

S'il n'existe aucun ensemble critique, nous pouvons ajouter à A un arc sortant de R. Sinon soit X l'ensemble critique maximal, et soit $e = (x, y)$ un arc tel que $x \in R \setminus X$ et $y \in V(G) \setminus (R \cup X)$ (voir figure 6.4). Un tel arc existe puisque

$$|\delta^+_{G-E(A)}(R \cup X)| \ = \ |\delta^+_G(R \cup X)| \ \geq \ k \ > \ k - 1 \ = \ |\delta^+_{G-E(A)}(X)|.$$

Ajoutons alors e à A. $A + e$ est clairement une arborescence enracinée en r. Il nous reste à montrer que la condition (6.2) est toujours vérifiée.

Supposons qu'il existe Y avec $r \in Y \subset V(G)$ et $|\delta_G^+(Y) \setminus E(A + e)| < k - 1$. On a $x \in Y$, $y \notin Y$, et $|\delta_G^+(Y) \setminus E(A)| = k - 1$. Le lemme 2.1(a) implique alors

$$\begin{aligned} k - 1 + k - 1 \ &= \ |\delta^+_{G-E(A)}(X)| + |\delta^+_{G-E(A)}(Y)| \\ &\geq \ |\delta^+_{G-E(A)}(X \cup Y)| + |\delta^+_{G-E(A)}(X \cap Y)| \\ &\geq \ k - 1 + k - 1 \ , \end{aligned}$$

parce que $r \in X \cap Y$ et $y \in V(G) \setminus (X \cup Y)$. Donc ces relations sont satisfaites avec égalité ; en particulier $|\delta^+_{G-E(A)}(X \cup Y)| = k - 1$. Puisque $y \in V(G) \setminus (X \cup Y \cup R)$, $X \cup Y$ est critique. Puisque $x \in Y \setminus X$, cela contredit la maximalité de X. □

La preuve précédente est due à Lovász [1976]. Une généralisation des théorèmes 6.16 et 6.17 a été trouvée par Frank [1978]. Une bonne caractérisation du problème de l'empilement d'arborescences avec des racines arbitraires est fournie par le théorème suivant que nous citons sans preuve :

Théorème 6.18. (Frank [1979]) *Un graphe orienté G contient k arborescences couvrantes arc-disjointes si et seulement si*

$$\sum_{i=1}^{p} |\delta^-(X_i)| \;\geq\; k(p-1)$$

pour toute collection de sous-ensembles non vides disjoints deux à deux $X_1, \ldots, X_p \subseteq V(G)$.

Un autre problème est de déterminer le nombre de forêts nécessaires pour couvrir un graphe. La réponse est donnée par le théorème suivant :

Théorème 6.19. (Nash-Williams [1964]) *L'ensemble des arêtes d'un graphe non orienté G est l'union de k forêts si et seulement si $|E(G[X])| \leq k(|X|-1)$ pour tout $\emptyset \neq X \subseteq V(G)$.*

Preuve. La condition est nécessaire puisque aucune forêt ne peut contenir plus que $|X|-1$ arêtes dont les deux extrémités sont dans X. Pour montrer qu'elle est suffisante, supposons que $|E(G[X])| \leq k(|X|-1)$ pour tout $\emptyset \neq X \subseteq V(G)$, et soit $F = (F_1, \ldots, F_k)$ un k-uplet de forêts disjointes dans G tel que $|E(F)| = \left|\bigcup_{i=1}^{k} E(F_i)\right|$ soit maximum. Montrons que $E(F) = E(G)$: supposons qu'il existe une arête $e \in E(G) \setminus E(F)$. Par le lemme 6.15, il existe un ensemble $X \subseteq V(G)$ avec $e \subseteq X$ tel que $F_i[X]$ soit connexe pour tout i. En particulier,

$$|E(G[X])| \;\geq\; \left|\{e\} \,\dot\cup \bigcup_{i=1}^{k} E(F_i[X])\right| \;\geq\; 1 + k(|X|-1),$$

ce qui contredit notre hypothèse. □

L'exercice 22 donne une version orientée de ce résultat. L'exercice 18 du chapitre 13 propose une généralisation des théorèmes 6.16 et 6.19 aux matroïdes.

Exercices

1. Montrer le théorème de Cayley, qui affirme que K_n a n^{n-2} arbres couvrants, en montrant que l'étiquetage suivant définit une bijection entre les arbres couvrants dans K_n et les vecteurs dans $\{1, \ldots, n\}^{n-2}$: pour un arbre T avec $V(T) = \{1, \ldots, n\}$, $n \geq 3$, soit v la feuille de plus petit indice et soit a_1 le voisin de v. Définissons récursivement $a(T) := (a_1, \ldots, a_{n-2})$, où $(a_2, \ldots, a_{n-2}) = a(T - v)$.
   (Cayley [1889], Prüfer [1918])

2. Soient (V, T_1) et (V, T_2) deux arbres définis sur le même ensemble V. Montrer que quelle que soit l'arête $e \in T_1$ il existe une arête $f \in T_2$ telle que $(V, (T_1 \setminus \{e\}) \cup \{f\})$ et $(V, (T_2 \setminus \{f\}) \cup \{e\})$ sont des arbres.
3. Soit un graphe G non orienté muni de poids $c : E(G) \to \mathbb{R}$ et soit v un sommet de $V(G)$; nous cherchons un arbre couvrant de poids minimum G n'ayant pas v comme feuille. Peut-on résoudre ce problème en temps polynomial ?
4. Trouver l'ensemble des arêtes d'un graphe non orienté G muni de poids $c : E(G) \to \mathbb{R}$ qui appartiennent à un arbre couvrant de poids minimum de G. Montrer que ce problème peut se résoudre en un temps $O(mn)$.
5. Soit un graphe G non orienté avec des poids $c : E(G) \to \mathbb{R}$. On cherche le sous-graphe couvrant de poids minimum. Peut-on résoudre ce problème en temps polynomial ?
6. Considérons l'algorithme suivant (appelé quelquefois ALGORITHME GLOUTON-PIRE-SORTI, voir paragraphe 13.4). Examiner les arêtes par poids décroissant. Supprimer l'arête courante sauf si c'est un pont. Cet algorithme résout-il le PROBLÈME DE L'ARBRE COUVRANT MINIMUM ?
7. Considérons l'algorithme suivant de «coloration». Initialement aucun arc n'est coloré. Appliquer ensuite les règles suivantes dans un ordre quelconque jusqu'à coloration complète de toutes les arêtes :
   • règle bleue : choisir une coupe ne contenant aucune arête bleue. Parmi les arêtes non colorées de cette coupe, en sélectionner une de coût minimum et la colorer en bleu ;
   • règle rouge : choisir un cycle ne contenant aucune arête rouge. Parmi les arêtes non colorées de ce cycle, en sélectionner une de coût maximum et la colorer en rouge.
   Montrer que tant qu'il existe une arête non colorée, une de ces deux règles s'applique. Montrer également que l'algorithme maintient à toute étape la propriété suivante : il existe toujours un arbre couvrant optimum contenant toutes les arêtes bleues et aucune arête rouge. (Cet algorithme résout donc le PROBLÈME DE L'ARBRE COUVRANT MINIMUM.) Observons que l'ALGORITHME DE KRUSKAL et celui de PRIM en sont des cas particuliers.
   (Tarjan [1983])
8. Supposons qu'on cherche un arbre couvrant T dans un graphe non orienté de telle sorte que le poids maximum d'une arête de T soit le plus petit possible. Comment trouver un tel arbre ?
9. Si $V \subset \mathbb{R}^2$ est un ensemble de points, le diagramme de Voronoï est formé des régions

$$P_v := \left\{ x \in \mathbb{R}^2 : ||x - v||_2 = \min_{w \in V} ||x - w||_2 \right\}$$

pour $v \in V$. La triangulation de Delaunay de V est le graphe

$$(V, \{\{v, w\} \subseteq V,\, v \neq w,\, |P_v \cap P_w| > 1\}) .$$

Un arbre couvrant minimum de V est un arbre T avec $V(T) = V$ de longueur $\sum_{\{v,w\}\in E(T)} ||v - w||_2$ minimum. Montrer que tout arbre couvrant minimum est un sous-graphe de la triangulation de Delaunay.
Note : utilisant le fait qu'une triangulation de Delaunay peut se calculer en un temps $O(n \log n)$ (où $n = |V|$; voir par exemple Fortune [1987], Knuth [1992]), cela conduit à un algorithme en $O(n \log n)$ pour le PROBLÈME DE L'ARBRE COUVRANT MINIMUM pour un ensemble de points du plan.
(Shamos et Hoey [1975] ; voir aussi Zhou, Shenoy et Nicholls [2002])

10. Peut-on savoir en temps linéaire si un graphe contient une arborescence couvrante ?
Indication : pour trouver une racine, partir d'un sommet quelconque et parcourir, aussi longtemps que possible, les arcs en sens inverse. Quand un circuit est détecté, le contracter.

11. Peut-on trouver en temps linéaire une ramification de cardinalité maximum ?
Indication : trouver d'abord les composantes fortement connexes.

12. Le PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM se réduit au PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM par la proposition 6.8. On peut cependant le résoudre directement en adaptant l'ALGORITHME DE RAMIFICATION D'EDMONDS. Montrer comment.

13. Montrer que le polytope des arbres couvrants dans un graphe non orienté G (voir théorème 6.12) avec $n := |V(G)|$ est en général strictement inclus dans le polytope

$$\left\{ x \in [0,1]^{E(G)} : \sum_{e\in E(G)} x_e = n - 1, \sum_{e\in\delta(X)} x_e \geq 1 \text{ pour } \emptyset \subset X \subset V(G) \right\}.$$

Indication : pour montrer que ce polytope n'est pas entier, étudier le graphe de la figure 6.5 (les nombres représentent les poids des arcs).
(Magnanti et Wolsey [1995])

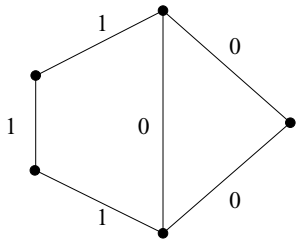


Figure 6.5.

∗ 14. Nous avons vu dans l'exercice 13 que les contraintes de coupe ne suffisent pas à décrire le polytope des arbres couvrants. Cependant en considérant les multicoupes, nous obtenons une description complète : montrer que le polytope des

arbres couvrants d'un graphe non orienté G avec $n := |V(G)|$ est l'ensemble des vecteurs $x \in [0,1]^{E(G)}$ tels que

$$\sum_{e \in E(G)} x_e = n-1; \sum_{e \in C} x_e \geq k-1 \text{ pour toute multicoupe } C = \delta(X_1, \ldots, X_k).$$

(Magnanti et Wolsey [1995])

15. Montrer que l'enveloppe convexe des vecteurs d'incidence des forêts dans un graphe non orienté G est le polytope

$$P := \left\{ x \in [0,1]^{E(G)} : \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ pour } \emptyset \neq X \subseteq V(G) \right\}.$$

Note : cette propriété implique le théorème 6.12 puisque $\sum_{e \in E(G[X])} x_e = |V(G)| - 1$ est un hyperplan support. C'est de plus un cas particulier du théorème 13.21.

∗ 16. Montrer que l'enveloppe convexe des vecteurs d'incidence des ramifications dans un graphe orienté G est l'ensemble des vecteurs $x \in [0,1]^{E(G)}$ tels que

$$\sum_{e \in E(G[X])} x_e \leq |X|-1 \text{ pour } \emptyset \neq X \subseteq V(G) \text{ et } \sum_{e \in \delta^-(v)} x_e \leq 1 \text{ pour } v \in V(G).$$

Note : cela est un cas particulier du théorème 14.13.

∗ 17. Soient G un graphe orienté et $r \in V(G)$. Montrer que chacun des deux polytopes

$$\left\{ x \in [0,1]^{E(G)} \quad : \quad x_e = 0 \ (e \in \delta^-(r)), \sum_{e \in \delta^-(v)} x_e = 1 \ (v \in V(G) \setminus \{r\}), \right.$$
$$\left. \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ pour } \emptyset \neq X \subseteq V(G) \right\}$$

et

$$\left\{ x \in [0,1]^{E(G)} \quad : \quad x_e = 0 \ (e \in \delta^-(r)), \sum_{e \in \delta^-(v)} x_e = 1 \ (v \in V(G) \setminus \{r\}), \right.$$
$$\left. \sum_{e \in \delta^+(X)} x_e \geq 1 \text{ pour } r \in X \subset V(G) \right\}$$

est égal à l'enveloppe convexe des vecteurs d'incidence de toutes les arborescences enracinées en r.

18. Soient G un graphe orienté et $r \in V(G)$. Montrer que G est l'union de k arborescences enracinées en r si et seulement si le graphe non orienté associé est l'union disjointe de k arbres couvrants et $|\delta^-(x)| = k$ pour tout $x \in V(G) \setminus \{r\}$. (Edmonds)

19. Soient G un graphe orienté et $r \in V(G)$. Supposons que G contienne k chemins arc-disjoints de r à tout autre sommet, mais que la suppression d'un arc détruise cette propriété. Montrer que tout sommet de G sauf r a exactement k arcs entrants.
Indication : utiliser le théorème 6.17.

∗ 20. Résoudre l'exercice 19 sans utiliser le théorème 6.17. Formuler et prouver une version sommet-disjoint.
Indication : si un sommet v a plus que k arcs entrants, considérons k chemins arc-disjoints de r à v. Montrer qu'un arc entrant en v qui n'est pas utilisé par un de ces chemins peut être supprimé.

21. Proposer un algorithme polynomial pour trouver un ensemble maximum d'arborescences couvrantes arc-disjointes (enracinées en r) dans un graphe orienté G.
Note : l'algorithme le plus efficace est celui de Gabow [1995] ; voir aussi Gabow et Manu [1998].

22. Montrer que les arcs d'un graphe orienté G peuvent être couverts par k ramifications si et seulement si les deux conditions suivantes sont vérifiées :
(a) $|\delta^-(v)| \leq k$ pour tout $v \in V(G)$.
(b) $|E(G[X])| \leq k(|X| - 1)$ pour tout $X \subseteq V(G)$.
Indication : utiliser le théorème 6.17.
(Frank [1979])

Références

Littérature générale :

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. [1993] : Network Flows. Prentice-Hall, Englewood Cliffs 1993, Chapter 13

Balakrishnan, V.K. [1995] : Network Optimization. Chapman and Hall, London 1995, Chapter 1

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. [2001] : Introduction to Algorithms. Second Edition. MIT Press, Cambridge 2001, Chapter 23

Gondran, M., Minoux, M. [1984] : Graphs and Algorithms. Wiley, Chichester 1984, Chapter 4

Magnanti, T.L., Wolsey, L.A. [1995] : Optimal trees. In : Handbooks in Operations Research and Management Science ; Volume 7 : Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995, pp. 503–616

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 50–53

Tarjan, R.E. [1983] : Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Chapter 6

Wu, B.Y., Chao, K.-M. [2004] : Spanning Trees and Optimization Problems. Chapman & Hall/CRC, Boca Raton 2004

Références citées :

Bock, F.C. [1971] : An algorithm to construct a minimum directed spanning tree in a directed network. In : Avi-Itzak, B. (Ed.) : Developments in Operations Research. Gordon and Breach, New York 1971, 29–44

Borůvka, O. [1926a] : O jistém problému minimálním. Práca Moravské Přírodovědecké Spolnečnosti 3 (1926), 37–58

Borůvka, O. [1926b] : Příspevěk k řešení otázky ekonomické stavby. Elektrovodních sítí. Elektrotechnicky Obzor 15 (1926), 153–154

Cayley, A. [1889] : A theorem on trees. Quarterly Journal on Mathematics 23 (1889), 376–378

Chazelle, B. [2000] : A minimum spanning tree algorithm with inverse-Ackermann type complexity. Journal of the ACM 47 (2000), 1028–1047

Cheriton, D., Tarjan, R.E. [1976] : Finding minimum spanning trees. SIAM Journal on Computing 5 (1976), 724–742

Chu, Y., Liu, T. [1965] : On the shortest arborescence of a directed graph. Scientia Sinica 4 (1965), 1396–1400 ; Mathematical Review 33, # 1245

Diestel, R. [1997] : Graph Theory. Springer, New York 1997

Dijkstra, E.W. [1959] : A note on two problems in connexion with graphs. Numerische Mathematik 1 (1959), 269–271

Dixon, B., Rauch, M., Tarjan, R.E. [1992] : Verification and sensitivity analysis of minimum spanning trees in linear time. SIAM Journal on Computing 21 (1992), 1184–1192

Edmonds, J. [1967] : Optimum branchings. Journal of Research of the National Bureau of Standards B 71 (1967), 233–240

Edmonds, J. [1970] : Submodular functions, matroids and certain polyhedra. In : Combinatorial Structures and Their Applications ; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schonheim, eds.), Gordon and Breach, New York 1970, pp. 69–87

Edmonds, J. [1973] : Edge-disjoint branchings. In : Combinatorial Algorithms (R. Rustin, ed.), Algorithmic Press, New York 1973, pp. 91–96

Fortune, S. [1987] : A sweepline algorithm for Voronoi diagrams. Algorithmica 2 (1987), 153–174

Frank, A. [1978] : On disjoint trees and arborescences. In : Algebraic Methods in Graph Theory ; Colloquia Mathematica ; Soc. J. Bolyai 25 (L. Lovász, V.T. Sós, eds.), North-Holland, Amsterdam 1978, pp. 159–169

Frank, A. [1979] : Covering branchings. Acta Scientiarum Mathematicarum (Szeged) 41 (1979), 77–82

Fredman, M.L., Tarjan, R.E. [1987] : Fibonacci heaps and their uses in improved network optimization problems. Journal of the ACM 34 (1987), 596–615

Fredman, M.L., Willard, D.E. [1994] : Trans-dichotomous algorithms for minimum spanning trees and shortest paths. Journal of Computer and System Sciences 48 (1994), 533–551

Fulkerson, D.R. [1974] : Packing rooted directed cuts in a weighted directed graph. Mathematical Programming 6 (1974), 1–13

Gabow, H.N. [1995] : A matroid approach to finding edge connectivity and packing arborescences. Journal of Computer and System Sciences 50 (1995), 259–273

Gabow, H.N., Galil, Z., Spencer, T. [1989] : Efficient implementation of graph algorithms using contraction. Journal of the ACM 36 (1989), 540–572

Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E. [1986] : Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica 6 (1986), 109–122

Gabow, H.N., Manu, K.S. [1998] : Packing algorithms for arborescences (and spanning trees) in capacitated graphs. Mathematical Programming B 82 (1998), 83–109

Jarník, V. [1930] : O jistém problému minimálním. Práca Moravské Přírodovědecké Společnosti 6 (1930), 57–63

Karger, D., Klein, P.N., Tarjan, R.E. [1995] : A randomized linear-time algorithm to find minimum spanning trees. Journal of the ACM 42 (1995), 321–328

Karp, R.M. [1972] : A simple derivation of Edmonds' algorithm for optimum branchings. Networks 1 (1972), 265–272

King, V. [1995] : A simpler minimum spanning tree verification algorithm. Algorithmica 18 (1997), 263–270

Knuth, D.E. [1992] : Axioms and hulls ; LNCS 606. Springer, Berlin 1992

Korte, B., Nešetřil, J. [2001] : Vojtěch Jarník's work in combinatorial optimization. Discrete Mathematics 235 (2001), 1–17

Kruskal, J.B. [1956] : On the shortest spanning subtree of a graph and the travelling salesman problem. Proceedings of the AMS 7 (1956), 48–50

Lovász, L. [1976] : On two minimax theorems in graph. Journal of Combinatorial Theory B 21 (1976), 96–103

Nash-Williams, C.S.J.A. [1961] : Edge-disjoint spanning trees of finite graphs. Journal of the London Mathematical Society 36 (1961), 445–450

Nash-Williams, C.S.J.A. [1964] : Decompositions of finite graphs into forests. Journal of the London Mathematical Society 39 (1964), 12

Nešetřil, J., Milková, E., Nešetřilová, H. [2001] : Otakar Borůvka on minimum spanning tree problem. Translation of both the 1926 papers, comments, history. Discrete Mathematics 233 (2001), 3–36

Prim, R.C. [1957] : Shortest connection networks and some generalizations. Bell System Technical Journal 36 (1957), 1389–1401

Prüfer, H. [1918] : Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. 27 (1918), 742–744

Shamos, M.I., Hoey, D. [1975] : Closest-point problems. Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science (1975), 151–162

Tarjan, R.E. [1975] : Efficiency of a good but not linear set union algorithm. Journal of the ACM 22 (1975), 215–225

Tutte, W.T. [1961] : On the problem of decomposing a graph into n connected factor. Journal of the London Mathematical Society 36 (1961), 221–230

Zhou, H., Shenoy, N., Nicholls, W. [2002] : Efficient minimum spanning tree construction without Delaunay triangulation. Information Processing Letters 81 (2002), 271–276

Chapitre 7

Plus courts chemins

Un des problèmes d'optimisation combinatoire parmi les plus connus est celui de la recherche d'un plus court chemin entre deux sommets donnés d'un graphe :

PROBLÈME DU PLUS COURT CHEMIN	
Instance	Un graphe orienté G, des poids $c : E(G) \to \mathbb{R}$ et deux sommets $s, t \in V(G)$.
Tâche	Trouver un plus court chemin P de s à t, c.-à-d. un chemin de poids $c(E(P))$ minimum, ou décider que t n'est pas connecté à s.

Ce problème a de nombreuses applications pratiques. Il apparaît souvent, de même que le PROBLÈME DE L'ARBRE COUVRANT MINIMUM, en tant que sous-problème dans l'étude de problèmes d'optimisation combinatoire plus difficiles.

Ce problème n'est pas simple à résoudre si les poids sont quelconques. Par exemple, quand les poids valent -1, les chemins de s à t de poids $1 - |V(G)|$ sont les chemins hamiltoniens de s à t. Or décider si un tel chemin existe est un problème difficile (voir exercice 14(b) du chapitre 15).

Le problème devient plus facile quand les circuits ont un poids total non négatif, ce qui est le cas si les poids des arcs sont non négatifs :

Définition 7.1. *Soit G un graphe orienté (resp. non orienté) muni de poids $c : E(G) \to \mathbb{R}$. c est dit* **conservatif** *s'il n'existe pas de circuit (resp. de cycle) de poids total négatif.*

Nous étudierons des algorithmes de résolution du PROBLÈME DU PLUS COURT CHEMIN au paragraphe 7.1. Le premier autorise seulement des poids non négatifs tandis que le second s'applique quand les poids sont conservatifs. Ces algorithmes calculent en réalité les plus courts chemins de s à v pour tout $v \in V(G)$, sans que cette généralisation n'augmente le temps de calcul. Le problème de la recherche des distances entre toutes les paires de sommets sera étudié au paragraphe 7.2.

Puisque les circuits négatifs posent des problèmes, nous montrerons comment les détecter. Si aucun circuit négatif n'existe, on pourra facilement trouver un circuit

de poids total minimum. Un autre problème intéressant est celui de la recherche du circuit de poids moyen minimum. Nous verrons au paragraphe 7.3 que ce problème peut se résoudre de manière efficace en utilisant des techniques similaires.

Trouver les plus courtes chaînes dans des graphes non orientés est plus difficile sauf si les poids sont non négatifs : les arêtes de poids non négatifs peuvent être alors remplacées par deux arcs d'orientation opposée munis du même poids ; cela nous ramène au cas orienté. Cependant, cette construction ne s'applique pas pour les arcs de poids négatifs puisqu'elle crée des circuits négatifs. Le problème de la plus courte chaîne dans des graphes non orientés avec des poids conservatifs sera étudié au paragraphe 12.2 (corollaire 12.12).

Nous n'étudierons donc que les graphes orientés G que nous pourrons supposer connexes et simples ; dans un ensemble d'arcs parallèles nous ne garderons que l'arc de plus faible poids.

7.1 Plus courts chemins à partir d'une source

Tous les algorithmes de plus court chemin que nous présentons sont fondés sur l'observation suivante appelée quelquefois principe d'optimalité de Bellman, qui est le fondement de la programmation dynamique :

Proposition 7.2. *Soit G un graphe orienté avec des poids conservatifs $c : E(G) \to \mathbb{R}$, soit $k \in \mathbb{N}$, et soient s, w deux sommets. Soit P un plus court chemin parmi les chemins de s à w ayant au plus k arcs, et soit $e = (v, w)$ son dernier arc. Alors $P_{[s,v]}$ (c.-à-d. P sans son dernier arc e) est un plus court chemin parmi les chemins de s à v ayant au plus $k - 1$ arcs.*

Preuve. Si Q est un chemin de s à v plus court que $P_{[s,v]}$ tel que $|E(Q)| \leq k-1$, alors $c(E(Q)) + c(e) < c(E(P))$. Si Q ne contient pas w, alors $Q + e$ est un chemin de s à w plus court que P ; sinon $Q_{[s,w]}$ a pour longueur $c(E(Q_{[s,w]})) = c(E(Q)) + c(e) - c(E(Q_{[w,v]} + e)) < c(E(P)) - c(E(Q_{[w,v]} + e)) \leq c(E(P))$, parce que $Q_{[w,v]} + e$ est un circuit, et parce que c est conservatif. Cela contredit dans les deux cas l'hypothèse que P est un plus court chemin de s à w avec au plus k arcs. □

Le même résultat s'applique aux graphes non orientés avec des poids non négatifs et aussi aux graphes sans circuit avec des poids quelconques. Cela conduit aux formules récursives : $\mathrm{dist}(s, s) = 0$ et $\mathrm{dist}(s, w) = \min\{\mathrm{dist}(s, v) + c((v, w)) : (v, w) \in E(G)\}$ pour $w \in V(G) \setminus \{s\}$, qui permettent de résoudre le PROBLÈME DU PLUS COURT CHEMIN pour les graphes sans circuit (exercice 6).

La proposition 7.2 permet également de comprendre pour quelle raison la plupart des algorithmes calculent les plus courts chemins de s vers tous les autres sommets. Quand on trouve un plus court chemin P de s à t on a déjà calculé le plus court chemin de s à v pour chaque sommet v de P. Comme on ne peut savoir à l'avance quels sommets appartiennent à P, il est naturel de calculer les plus courts

chemins de s à v pour tout v. Nous pouvons retrouver les chemins de s à v de manière efficace en stockant seulement le dernier arc de chaque chemin.

Nous allons considérer le cas où les poids sont non négatifs, c.-à-d. $c : E(G) \to \mathbb{R}_+$. Le PROBLÈME DU PLUS COURT CHEMIN peut se résoudre par BFS si tous les poids valent 1 (proposition 2.18). Pour les poids $c : E(G) \to \mathbb{N}$ on peut remplacer chaque arc e par un chemin de longueur $c(e)$ et de nouveau utiliser BFS. Mais cela conduit à la création d'un nombre exponentiel d'arcs ; en effet, la taille de l'input est $\Theta\left(n \log m + m \log n + \sum_{e \in E(G)} \log c(e)\right)$ ($n = |V(G)|$, $m = |E(G)|$).

Une bien meilleure idée est d'utiliser l'algorithme suivant, dû à Dijkstra [1959]. C'est une méthode semblable à celle de l'ALGORITHME DE PRIM pour le PROBLÈME DE L'ARBRE COUVRANT MINIMUM (voir paragraphe 6.1).

ALGORITHME DE DIJKSTRA

Input Un graphe orienté G, des poids $c : E(G) \to \mathbb{R}_+$ et $s \in V(G)$.

Output Les plus courts chemins de s à tout $v \in V(G)$ et leurs longueurs. Plus précisément, nous obtenons $l(v)$ et $p(v)$ pour tout $v \in V(G) : l(v)$ est la longueur d'un plus court chemin de s à v, consistant en un plus court chemin de s à $p(v)$ et de l'arc $(p(v), v)$. Si v n'est pas connecté à s, alors $l(v) = \infty$ et $p(v)$ n'est pas défini.

① $l(s) := 0$. $l(v) := \infty$ pour tout $v \in V(G) \setminus \{s\}$.
$R := \emptyset$.

② Trouver un sommet $v \in V(G) \setminus R$ tel que $l(v) = \min_{w \in V(G) \setminus R} l(w)$.

③ $R := R \cup \{v\}$.

④ **For** tout $w \in V(G) \setminus R$ tel que $(v, w) \in E(G)$ **do** :
If $l(w) > l(v) + c((v, w))$ **then**
poser $l(w) := l(v) + c((v, w))$ et $p(w) := v$.

⑤ **If** $R \neq V(G)$ **then go to** ②.

Théorème 7.3. (Dijkstra [1959]) *L'*ALGORITHME DE DIJKSTRA *répond correctement.*

Preuve. Nous allons montrer que les conditions suivantes sont toujours vérifiées :

(a) Pour tout $v \in V(G) \setminus \{s\}$ avec $l(v) < \infty$ $p(v) \in R$, $l(p(v)) + c((p(v), v)) = l(v)$, et la suite $v, p(v), p(p(v)), \ldots$ contient s.

(b) Pour tout $v \in R : l(v) = \mathrm{dist}_{(G,c)}(s, v)$.

Ces conditions sont trivialement vérifiées après ①. $l(w)$ est abaissé à $l(v) + c((v, w))$ et $p(w)$ devient égal à v dans ④ uniquement si $v \in R$ et $w \notin R$. Comme la suite $v, p(v), p(p(v)), \ldots$ contient s mais aucun sommet en dehors de R, et en particulier ne contient pas w, (a) reste vérifié après ④.

(b) est évident pour $v = s$. Supposons que $v \in V(G) \setminus \{s\}$ soit ajouté à R dans ③, et qu'il existe un chemin P de s à v dans G plus court que $l(v)$. Soit y le premier

sommet de P qui appartient à $(V(G) \setminus R) \cup \{v\}$, et soit x le prédécesseur de y sur P. Puisque $x \in R$, nous avons par ④ et l'hypothèse d'induction :

$$\begin{aligned} l(y) \leq l(x) + c((x,y)) &= \operatorname{dist}_{(G,c)}(s,x) + c((x,y)) \\ &\leq c(E(P_{[s,y]})) \leq c(E(P)) < l(v), \end{aligned}$$

contredisant le choix de v dans ②. □

Le temps d'exécution est clairement $O(n^2)$. En utilisant un tas de Fibonacci nous pouvons améliorer la complexité :

Théorème 7.4. (Fredman et Tarjan [1987]) *L'*ALGORITHME DE DIJKSTRA *implémenté avec un tas de Fibonacci s'exécute en un temps $O(m + n \log n)$, où $n = |V(G)|$ et $m = |E(G)|$.*

Preuve. Appliquons le théorème 6.6 pour gérer $\{(v, l(v)) : v \in V(G) \setminus R,\ l(v) < \infty\}$. Alors ② et ③ sont des opérations de SUPPRESSION, tandis que la mise à jour de $l(w)$ dans ④ est une opération d'INSERTION si $l(w)$ est infini et sinon une opération de DÉCROISSANCE. □

C'est la meilleure complexité connue pour le PROBLÈME DU PLUS COURT CHEMIN avec des poids non négatifs. (Avec d'autres modèles de calcul, Fredman et Willard [1994], Thorup [2000] et Raman [1997] ont un peu amélioré cette borne.)

Si les poids sont des entiers restant dans une certaine fourchette, il existe un algorithme simple en temps linéaire (voir exercice 2). En général, on atteint des bornes en $O(m \log\log c_{\max})$ (Johnson [1982]) et $O\left(m + n\sqrt{\log c_{\max}}\right)$ (Ahuja *et al.* [1990]) pour des poids $c : E(G) \to \{0, \ldots, c_{\max}\}$. Cela a été amélioré par Thorup [2004] jusqu'à $O(m+n \log\log c_{\max})$ et $O(m+n \log\log n)$, mais ces bornes supposent des poids entiers, et l'algorithme n'est pas fortement polynomial.

Pour les graphes planaires orientés il existe un algorithme linéaire proposé par Henzinger *et al.* [1997]. Enfin signalons que Thorup [1999] a proposé un algorithme linéaire pour trouver un plus court chemin dans un graphe non orienté avec des poids entiers non négatifs. Voir également Pettie et Ramachandran [2005] ; cet article contient aussi d'autres références.

Étudions maintenant le cas général des poids conservatifs :

ALGORITHME DE MOORE-BELLMAN-FORD

Input	Un graphe orienté G, des poids conservatifs $c : E(G) \to \mathbb{R}$, et un sommet $s \in V(G)$.
Output	Les plus courts chemins de s à tout $v \in V(G)$ et leurs longueurs. Plus précisément, nous obtenons $l(v)$ et $p(v)$ pour chaque $v \in V(G)$. $l(v)$ est la longueur d'un plus court chemin de s à v, consistant en un plus court chemin de s à $p(v)$ et de l'arc $(p(v), v)$. Si v n'est pas connecté s, alors $l(v) = \infty$ et $p(v)$ n'est pas défini.

① $l(s) := 0$ et $l(v) := \infty$ pour tout $v \in V(G) \setminus \{s\}$.

② **For** $i := 1$ **to** $n-1$ **do** :
  For chaque arc $(v,w) \in E(G)$ **do** :
    If $l(w) > l(v) + c((v,w))$ **then**
      $l(w) := l(v) + c((v,w))$ et $p(w) := v$.

Théorème 7.5. (Moore [1959], Bellman [1958], Ford [1956]) L'ALGORITHME DE MOORE-BELLMAN-FORD *répond correctement. Sa complexité est* $O(nm)$.

Preuve. La complexité $O(mn)$ est évidente. À chaque étape de l'algorithme, posons $R := \{v \in V(G) : l(v) < \infty\}$ et $F := \{(x,y) \in E(G) : x = p(y)\}$. Montrons que les conditions suivantes sont vérifiées même avec des poids non conservatifs :

(a) $l(y) \geq l(x) + c((x,y))$ pour tout $(x,y) \in F$.

(b) Si F contient un circuit C, le poids de C est négatif.

Pour montrer (a), remarquons que $l(y) = l(x) + c((x,y))$ quand $p(y)$ devient égal à x et observons que $l(x)$ n'augmente jamais.

Pour montrer (b), supposons qu'un circuit C dans F soit créé quand on arrive à l'instruction $p(y) := x$. Avant cette instruction, nous avions $l(y) > l(x) + c((x,y))$ et $l(w) \geq l(v) + c((v,w))$ pour tout $(v,w) \in E(C) \setminus \{(x,y)\}$ (par (a)). En sommant ces inégalités, on obtient que le poids total de C est négatif.

Utilisons maintenant l'hypothèse que les poids sont conservatifs. (b) implique alors que F est sans circuit. De plus, $x \in R \setminus \{s\}$ implique que $p(x) \in R$, et donc (R,F) est une arborescence enracinée en s.

Donc $l(x)$ est au moins égal à la longueur du chemin de s à x dans (R,F) pour tout $x \in R$ (à n'importe quelle étape de l'algorithme).

Montrons par induction que, après k itérations de l'algorithme, $l(x)$ est plus petit ou égal à la longueur d'un plus court chemin de s à x ayant au plus k arcs : soit P un plus court chemin de s à x ayant au plus k arcs et soit (w,x) le dernier arc de P. Par la proposition 7.2, $P_{[s,w]}$ est un plus court chemin de s à w ayant au plus $k-1$ arcs, et par l'hypothèse d'induction $l(w) \leq c(E(P_{[s,w]}))$ après $k-1$ itérations. Mais, à l'itération k, l'arc (w,x) est traité et on aura après $l(x) \leq l(w) + c((w,x)) \leq c(E(P))$.

Puisque aucun chemin ne contient plus que $n-1$ arcs, cela montre bien que l'algorithme répond correctement. □

Cet algorithme est actuellement le plus rapide algorithme fortement polynomial connu pour le PROBLÈME DU PLUS COURT CHEMIN (avec des poids conservatifs). Un algorithme de réduction d'échelle de Goldberg [1995] donne un temps de calcul $O\left(\sqrt{n}m \log(|c_{\min}| + 2)\right)$ si les poids des arcs sont entiers et valent au moins $c_{\min}$. Pour les graphes planaires, Fakcharoenphol et Rao [2006] ont proposé un algorithme en $O(n \log^3 n)$.

Si G contient des circuits négatifs, on ne connaît pas d'algorithme polynomial (le problème devient *NP*-difficile ; voir l'exercice 14(b) du chapitre 15). La principale difficulté est que la proposition 7.2 est fausse quand les poids sont quelconques.

Il n'est pas simple de construire un chemin au lieu d'un parcours. S'il n'y a pas de circuit négatif, un plus court parcours est un chemin avec éventuellement des circuits de poids nul qui peuvent être éliminés. Au vu de cela , il est important de détecter les circuits négatifs. La notion suivante introduite par Edmonds et Karp [1972] sera utile :

Définition 7.6. *Soit G un graphe orienté avec des poids $c : E(G) \to \mathbb{R}$, et soit $\pi : V(G) \to \mathbb{R}$. Pour tout $(x, y) \in E(G)$, définissons le* **coût réduit** *de (x, y) par rapport à π par $c_\pi((x, y)) := c((x, y)) + \pi(x) - \pi(y)$. Si $c_\pi(e) \geq 0$ pour tout $e \in E(G)$, nous dirons que π est un* **potentiel réalisable**.

Théorème 7.7. *Soit G un graphe orienté muni de poids $c : E(G) \to \mathbb{R}$. (G, c) admet un potentiel réalisable si et seulement si les poids c sont conservatifs.*

Preuve. Si π est un potentiel réalisable, pour chaque circuit C :

$$0 \leq \sum_{e \in E(C)} c_\pi(e) = \sum_{e=(x,y) \in E(C)} (c(e) + \pi(x) - \pi(y)) = \sum_{e \in E(C)} c(e)$$

(les potentiels s'éliminent). Donc les poids c sont conservatifs.

D'autre part, si les poids c sont conservatifs, ajoutons un nouveau sommet s et les arcs (s, v) avec un coût nul pour tout $v \in V(G)$. Exécutons l'ALGORITHME DE MOORE-BELLMAN-FORD sur cette instance ; nous obtenons $l(v)$ pour tout $v \in V(G)$. $l(v)$ est la longueur d'un plus court chemin de s à v pour tout $v \in V(G)$, et nous avons donc $l(w) \leq l(v) + c((v, w))$ pour tout arc $(v, w) \in E(G)$. l est donc un potentiel réalisable. □

Ce résultat peut être considéré comme une application de la dualité en programmation linéaire ; voir l'exercice 8.

Corollaire 7.8. *Soit G un graphe orienté avec des poids $c : E(G) \to \mathbb{R}$; on peut trouver en un temps $O(nm)$ soit un potentiel réalisable, soit un circuit négatif.*

Preuve. Comme précédemment, ajoutons un nouveau sommet s et des arcs (s, v) de coût nul pour tout $v \in V(G)$. Nous exécutons une variante de l'ALGORITHME DE MOORE-BELLMAN-FORD sur cette instance : que c soit conservatif ou non, nous exécutons ① et ②. Nous obtenons les quantités $l(v)$ pour tout $v \in V(G)$. Si l est un potentiel réalisable, l'algorithme est terminé.

Sinon, choisissons un arc (v, w) pour lequel $l(w) > l(v) + c((v, w))$. Montrons alors que la suite $w, v, p(v), p(p(v)), \ldots$ contient un circuit. En effet observons que $l(v)$ a forcément été modifié à la dernière itération de ②. Donc $l(p(v))$ a été changé durant les deux dernières itérations, $l(p(p(v)))$ durant les trois dernières, et ainsi de suite. Puisque $l(s)$ ne change jamais, les $|V(G)|$ premiers éléments de la suite $w, v, p(v), p(p(v)), \ldots$ ne contiennent pas s ; ainsi un sommet apparaît deux fois dans cette suite.

Il existe donc un circuit dans $F := \{(x, y) \in E(G) : x = p(y)\} \cup \{(v, w)\}$ qui par (a) et (b) du théorème 7.5 a un poids total négatif. □

En pratique, il existe des méthodes plus efficaces pour trouver un circuit négatif ; voir Cherkassky et Goldberg [1999].

7.2 Plus courts chemins entre toutes les paires de sommets

Supposons que nous recherchions les plus courts chemins de s à t pour toutes les paires ordonnées de sommets (s,t) d'un graphe orienté :

PROBLÈME DU PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES	
Instance	Un graphe orienté G et des poids conservatifs $c : E(G) \to \mathbb{R}$.
Tâche	Trouver les nombres l_{st} et les sommets p_{st} pour tout $s,t \in V(G)$ avec $s \neq t$, tels que l_{st} soit la longueur d'un plus court chemin (s'il existe) de s à t ayant (p_{st},t) comme dernier arc.

Bien entendu on peut exécuter L'ALGORITHME DE MOORE-BELLMAN-FORD n fois pour chaque $s \in V(G)$ et obtenir un algorithme en $O(n^2m)$. Mais on peut faire mieux comme l'ont montré Bazaraa et Langley [1974] et Johnson [1977] :

Théorème 7.9. *Le* PROBLÈME DU PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES *peut être résolu en un temps* $O(mn+n^2 \log n)$ *(*$n = |V(G)|$, $m = |E(G)|$*).*

Preuve. Soit (G,c) une instance. Calculons d'abord un potentiel réalisable π, ce qui est possible en $O(nm)$ par le corollaire 7.8. Effectuons alors un calcul de plus court chemin pour tout $s \in V(G)$, à partir de s, en utilisant les coûts réduits c_π à la place de c. Pour chaque sommet t le plus court chemin de s à t est aussi un plus court chemin pour les poids c, puisque la longueur de chaque chemin de s à t est modifiée par la constante $\pi(s) - \pi(t)$. Puisque les coûts réduits sont non négatifs, nous pouvons nous servir à chaque fois de l'ALGORITHME DE DIJKSTRA. Le temps total de calcul est $O(mn + n(m + n \log n))$ par le théorème 7.4. □

La même idée sera utilisée au chapitre 9 (dans la preuve du théorème 9.12).

Pettie [2004] a donné une borne en $O(mn + n^2 \log\log n)$; c'est la meilleure borne connue actuellement. Pour les graphes denses avec des poids non négatifs, la borne $O(n^3 \log^3 \log n / \log^2 n)$ proposée par Chan's [2007] est légèrement meilleure. Si les poids sont des petits entiers positifs, on peut améliorer cette borne en utilisant la multiplication matricielle rapide ; voir Zwick [2002].

La solution du PROBLÈME DU PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES permet aussi de calculer la fermeture métrique :

Définition 7.10. *Soit un graphe* G *(orienté ou non) avec des poids conservatifs* $c : E(G) \to \mathbb{R}$*. La* **fermeture métrique** *de* (G,c) *est la paire* $(\bar{G},\bar{c})$*, où* $\bar{G}$ *est le graphe simple défini sur* $V(G)$ *qui contient un arc ou une arête* $e = (x,y)$ *pour* $x,y \in V(G)$ *avec* $x \neq y$*, de poids* $\bar{c}(e) = \operatorname{dist}_{(G,c)}(x,y)$ *si et seulement si* y *est connecté à* x *dans* G*.*

Corollaire 7.11. *Soit* G *un graphe orienté muni de poids conservatifs* $c : E(G) \to \mathbb{R}$*, ou un graphe non orienté avec des poids non négatifs* $c : E(G) \to \mathbb{R}_+$*. Alors, la fermeture métrique de* (G,c) *peut se calculer en un temps* $O(mn + n^2 \log n)$*.*

Preuve. Si G est non orienté, nous remplaçons chaque arête par une paire d'arcs de directions opposées. Puis nous résolvons l'instance du PROBLÈME DU PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES. □

Le reste de ce paragraphe est consacré à l'ALGORITHME DE FLOYD-WARSHALL, qui est un autre algorithme en $O(n^3)$ pour le PROBLÈME DU PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES. L'intérêt de cet algorithme est sa simplicité. Nous supposerons sans perte de généralité que les sommets sont numérotés $1, \ldots, n$.

ALGORITHME DE FLOYD-WARSHALL

Input Un graphe orienté G avec $V(G) = \{1, \ldots, n\}$ et des poids conservatifs $c : E(G) \to \mathbb{R}$.

Output Deux matrices $(l_{ij})_{1 \le i,j \le n}$ et $(p_{ij})_{1 \le i,j \le n}$ où l_{ij} est la longueur d'un plus court chemin de i à j, et (p_{ij}, j) le dernier arc d'un tel chemin (s'il existe).

① $l_{ij} := c((i,j))$ pour tout $(i,j) \in E(G)$.
$l_{ij} := \infty$ pour tout $(i,j) \in (V(G) \times V(G)) \setminus E(G)$ avec $i \neq j$.
$l_{ii} := 0$ pour tout i.
$p_{ij} := i$ pour toute paire $i, j \in V(G)$.

② **For** $j := 1$ **to** n **do** :
For $i := 1$ **to** n **do** : **If** $i \neq j$ **then** :
For $k := 1$ **to** n **do** : **If** $k \neq j$ **then** :
If $l_{ik} > l_{ij} + l_{jk}$ **then** $l_{ik} := l_{ij} + l_{jk}$ et $p_{ik} := p_{jk}$.

Théorème 7.12. (Floyd [1962], Warshall [1962]) *L'*ALGORITHME DE FLOYD-WARSHALL *répond correctement. Sa complexité est* $O(n^3)$.

Preuve. La borne de complexité est évidente. Montrons que quand la première boucle a été effectuée pour $j = 1, 2, \ldots, j_0$, la variable l_{ik}, pour tout i et k, représente la longueur d'un plus court chemin de i à k dont tous les sommets intermédiaires appartiennent à l'ensemble $\{1, \ldots, j_0\}$, (p_{ik}, k) étant le dernier arc de ce chemin.

Raisonnons par induction sur $j_0 = 0, \ldots, n$. Cette propriété est vraie pour $j_0 = 0$ par ① ; pour $j_0 = n$, elle impliquera que l'algorithme répond correctement.

Supposons qu'elle soit vraie pour $j_0 \in \{0, \ldots, n-1\}$, et montrons qu'elle est vraie pour $j_0 + 1$. Pour tout i et k, quand la première boucle s'exécute pour $j = j_0 + 1$, l_{ik} (qui, par induction, est la longueur d'un plus court chemin de i à k dont tous les sommets intermédiaires appartiennent à l'ensemble $\{1, \ldots, j_0\}$) est remplacé par $l_{i,j_0+1} + l_{j_0+1,k}$ si cette valeur est plus petite. Il suffit de démontrer que le chemin P de i à $(j_0 + 1)$ et le chemin Q de $(j_0 + 1)$ à k n'ont pas de sommet interne commun, ce qui impliquera que $P + Q$ est un chemin.

Supposons qu'il existe un sommet interne qui appartienne à P et Q. Le parcours orienté $P + Q$ de i à k est l'union de circuits de poids total non négatifs (puisque les poids sont conservatifs) et d'un chemin de i à k qui ne contient pas $j_0 + 1$ et dont

tous les sommets intermédiaires sont dans $\{1, \ldots, j_0\}$. Mais le coût de ce chemin serait plus faible que l_{ik} calculé avant l'implémentation de la première boucle pour $j_0 + 1$, ce qui est une contradiction. □

On peut utiliser l'ALGORITHME DE FLOYD-WARSHALL (comme l'ALGORITHME DE MOORE-BELLMAN-FORD) pour détecter un circuit négatif (exercice 11).

Le PROBLÈME DE LA PLUS COURTE CHAÎNE ENTRE TOUTES LES PAIRES dans les graphes non orientés avec poids conservatifs est plus ardu ; voir théorème 12.13.

7.3 Circuit moyen minimum

On peut facilement trouver un circuit de poids total minimum dans un graphe orienté, en se servant des algorithmes de plus court chemin précédents (voir exercice 12). Un autre problème consiste à trouver le circuit de poids moyen minimum :

PROBLÈME DU CIRCUIT MOYEN MINIMUM	
Instance	Un graphe orienté G, des poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver un circuit C dont le poids moyen $\frac{c(E(C))}{\lvert E(C)\rvert}$ est minimum, ou décider que G est sans circuit.

Dans ce paragraphe nous allons montrer comment résoudre ce problème par programmation dynamique, comme pour les algorithmes de plus court chemin. Nous pouvons supposer G fortement connexe ; sinon nous pouvons identifier les composantes fortement connexes en temps linéaire (théorème 2.19) et résoudre le problème séparément dans chaque composante fortement connexe. Pour le théorème min-max suivant, il suffit de supposer qu'il existe un sommet s qui est racine du graphe. Nous considérerons ici des parcours (avec répétition éventuelle de sommets et d'arcs), et non pas nécessairement des chemins.

Théorème 7.13. (Karp [1978]) *Soit G un graphe orienté muni de poids $c : E(G) \to \mathbb{R}$. Soit $s \in V(G)$ tel que chaque sommet soit connecté à s. Pour $x \in V(G)$ et $k \in \mathbb{Z}_+$ soit*

$$F_k(x) := \min \left\{ \sum_{i=1}^{k} c((v_{i-1}, v_i)) : v_0 = s,\ v_k = x,\ (v_{i-1}, v_i) \in E(G) \text{ pour tout } i \right\}$$

le poids minimum d'un parcours orienté de longueur k de s à x (et ∞ si aucun parcours de ce type existe). Soit $\mu(G, c)$ le poids moyen minimum d'un circuit de G (et $\mu(G, c) = \infty$ si G est sans circuit). Alors

$$\mu(G, c) = \min_{x \in V(G)} \max_{\substack{0 \le k \le n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k}.$$

Preuve. Si G est sans circuit, alors $F_n(x) = \infty$ pour tout $x \in V(G)$ et le théorème est vrai. Supposons donc $\mu(G,c) < \infty$.

Montrons d'abord que si $\mu(G,c) = 0$,

$$\min_{x \in V(G)} \max_{\substack{0 \le k \le n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n-k} = 0.$$

Soit G un graphe orienté avec $\mu(G,c) = 0$. G ne contient pas de circuit négatif. Puisque c est conservatif, $F_n(x) \ge \mathrm{dist}_{(G,c)}(s,x) = \min_{0 \le k \le n-1} F_k(x)$; donc

$$\max_{\substack{0 \le k \le n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n-k} \ge 0.$$

Montrons que l'égalité est satisfaite pour un sommet x : $F_n(x) = \mathrm{dist}_{(G,c)}(s,x)$. Soit C un circuit quelconque de G de poids nul, et soit $w \in V(C)$. Considérons le parcours P constitué d'un plus court chemin de s à w suivi par n répétitions de C. Soit P' le parcours orienté constitué des n premiers arcs de P, et soit x le dernier sommet de P'. Puisque P est un parcours orienté de poids minimum de s à w, tout sous-parcours initial, en particulier P', est un parcours de poids minimum. Donc $F_n(x) = c(E(P')) = \mathrm{dist}_{(G,c)}(s,x)$.

Ayant prouvé le théorème dans le cas $\mu(G,c) = 0$, étudions maintenant le cas général. L'addition d'une constante à tous les poids modifie $\mu(G,c)$ et

$$\min_{x \in V(G)} \max_{\substack{0 \le k \le n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n-k}$$

d'une même quantité égale à cette constante. En prenant cette constante égale à $-\mu(G,c)$ nous sommes ramenés au cas $\mu(G,c) = 0$. □

Ce théorème conduit à l'algorithme suivant :

ALGORITHME DU CIRCUIT MOYEN MINIMUM	
Input	Un graphe orienté G, des poids $c : E(G) \to \mathbb{R}$.
Output	Un circuit C de poids moyen minimum ou l'indication que G est sans circuit.

① Ajouter à G un sommet s et les arcs (s,x) avec $c((s,x)) := 0$ pour tout $x \in V(G)$.

② $n := |V(G)|$, $F_0(s) := 0$, $F_0(x) := \infty$ pour tout $x \in V(G) \setminus \{s\}$.

③ **For** $k := 1$ **to** n **do** :
  For tout $x \in V(G)$ **do** :
    $F_k(x) := \infty$.
    For tout $(w,x) \in \delta^-(x)$ **do** :
      If $F_{k-1}(w) + c((w,x)) < F_k(x)$ **then** :
        $F_k(x) := F_{k-1}(w) + c((w,x))$ et $p_k(x) := w$.

④ **If** $F_n(x) = \infty$ pour tout $x \in V(G)$ **then stop** (G est sans circuit).

⑤ Soit x un sommet tel que $\max_{\substack{0 \le k \le n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n-k}$ soit minimum.

⑥ Soit C un circuit quelconque contenu dans le chemin défini par $p_n(x), p_{n-1}(p_n(x)), p_{n-2}(p_{n-1}(p_n(x))), \ldots$.

Corollaire 7.14. (Karp [1978]) *L'*ALGORITHME DU CIRCUIT MOYEN MINIMUM *répond correctement. Son temps d'exécution est en* $O(nm)$.

Preuve. ① ne crée pas de nouveau circuit dans G, mais permet d'appliquer le théorème 7.13. Il est évident que ② et ③ fournissent les valeurs exactes de $F_k(x)$. Donc si l'algorithme s'arrête en ④, G est sans circuit.

Considérons l'instance (G, c'), où $c'(e) := c(e) - \mu(G, c)$ pour tout $e \in E(G)$. Sur cette instance, l'algorithme se déroule de la même manière que sur l'instance (G, c), la seule différence étant que les valeurs F sont changées en $F'_k(x) = F_k(x) - k\mu(G, c)$. Par le choix de x dans ⑤, le théorème 7.13 et $\mu(G, c') = 0$, nous avons $F'_n(x) = \min_{0 \le k \le n-1} F'_k(x)$. Donc tout parcours orienté de s à x ayant n arcs et de longueur $F'_n(x)$ dans (G, c') est constitué d'un chemin de s à x et de l'addition d'un ou de plusieurs circuits de poids zéro. Ces circuits ont un poids moyen $\mu(G, c)$ dans (G, c).

Tout circuit extrait d'un parcours de longueur n de s à x (où x est le sommet choisi en ⑤) et de poids minimum est un circuit de poids moyen minimum. Un tel circuit est choisi en ⑥.

Le temps de calcul est déterminé par ③ qui nécessite un temps $O(nm)$. Notons que la complexité de ⑤ est seulement en $O(n^2)$. □

Cet algorithme ne s'étend pas au cas des graphes non orientés ; voir l'exercice 10 du chapitre 12.

Des algorithmes pour d'autres problèmes de ratio minimum ont été proposés par Megiddo [1979,1983] et Radzik [1993].

Exercices

1. Soit G un graphe non orienté (resp. orienté) muni de poids $c : E(G) \to \mathbb{Z}_+$, et soient $s, t \in V(G)$ tels que t soit connecté à s. Montrer que la longueur minimum d'une chaîne (resp. d'un chemin) de s à t est égale au nombre maximum de coupes (resp. coupes sortantes) séparant s et t (resp. t de s) de telle sorte que chaque arête (resp. arc) e soit contenue dans au plus $c(e)$ de ces coupes.
2. Supposons que les poids soient des entiers compris entre 0 et une constante C. Peut-on implémenter l'ALGORITHME DE DIJKSTRA pour qu'il s'exécute en temps linéaire ?
   Indication : utiliser un tableau indicé par $0, \ldots, |V(G)| \cdot C$ pour mémoriser les

sommets par distance non décroissante.
(Dial [1969])

3. Soit un graphe orienté G muni de poids $c : E(G) \to \mathbb{R}_+$, et soient deux sommets $s, t \in V(G)$. Supposons qu'il existe un seul plus court chemin P de s à t. Peut-on trouver un deuxième plus court chemin de s à t dans G en temps polynomial ?
4. Modifier l'ALGORITHME DE DIJKSTRA afin de résoudre le problème du chemin de seuil minimum : soit G un graphe orienté avec des poids $c : E(G) \to \mathbb{R}$ et $s, t \in V(G)$; trouver un chemin de s à t dont le maximum des coûts des arcs soit le plus petit possible.
5. Soit G un graphe orienté et $s, t \in V(G)$. Associons un nombre $r(e)$ (appelé fiabilité) à tout arc $e \in E(G)$, avec $0 \leq r(e) \leq 1$. La fiabilité d'un chemin P est définie comme étant le produit des fiabilités de ses arcs. Le problème consiste à trouver un chemin de s à t de fiabilité maximum.
   (a) Montrer comment on peut, en utilisant les logarithmes, se ramener à un PROBLÈME DE PLUS COURT CHEMIN.
   (b) Montrer comment résoudre ce problème en temps polynomial sans utiliser les logarithmes.
6. Soit G un graphe orienté sans circuit muni de poids $c : E(G) \to \mathbb{R}$ et soit $s, t, \in V(G)$. Montrer comment résoudre le problème du plus court chemin de s à t dans G en temps linéaire.
7. Soit G un graphe orienté sans circuit muni de poids $c : E(G) \to \mathbb{R}$ et soit $s, t, \in V(G)$. Montrer comment résoudre le problème du plus long chemin de s à t dans G en temps linéaire.
8. Montrer le théorème 7.7 en utilisant la dualité en PROGRAMMATION LINÉAIRE, en particulier le théorème 3.24.
9. Soit G un graphe orienté avec des poids conservatifs $c : E(G) \to \mathbb{R}$. Soient $s, t \in V(G)$ tel que t soit connecté à s. Montrer que le coût minimum d'un chemin de s à t de G est égal au maximum de $\pi(t) - \pi(s)$, où π est un potentiel réalisable de (G, c).
10. Soit G un graphe orienté, $V(G) = A \dot\cup B$ et $E(G[B]) = \emptyset$. De plus supposons que $|\delta(v)| \leq k$ pour tout $v \in B$. Soient $s, t \in V(G)$ et supposons que les coûts soient conservatifs $c : E(G) \to \mathbb{R}$. Montrer qu'on peut trouver un plus court chemin de s à t en un temps $O(|A|k|E(G)|)$, et en un temps $O(|A|^2)$ si c est non négatif.
    (Orlin [1993])
11. Supposons que nous appliquions l'ALGORITHME DE FLOYD-WARSHALL sur une instance (G, c) avec des poids arbitraires $c : E(G) \to \mathbb{R}$. Montrer que les quantités l_{ii} $(i = 1, \ldots, n)$ restent non négatives si et seulement si les coûts c sont conservatifs.
12. Soit un graphe orienté avec des coûts conservatifs ; montrer comment trouver en temps polynomial un circuit de poids minimum. Peut-on avoir une complexité

en $O(n^3)$?
Indication : modifier légèrement l'ALGORITHME DE FLOYD-WARSHALL.
Note : le problème quand les poids sont quelconques inclut le problème de l'existence d'un circuit hamiltonien dans un graphe orienté (et par conséquent est *NP*-difficile ; voir chapitre 15). La recherche d'un cycle de poids minimum (quand les poids sont conservatifs) dans un graphe non orienté sera abordé au paragraphe 12.2.

13. Soit G un graphe complet non orienté et $c : E(G) \to \mathbb{R}_+$. Montrer que (G, c) est sa propre fermeture métrique si et seulement si l'inégalité triangulaire : $c((x, y)) + c((y, z)) \geq c((x, z))$ pour tout triplet de sommets $x, y, z \in V(G)$ est vérifiée.

14. Les contraintes de délai d'une puce électronique peuvent se modéliser par un graphe orienté G muni de poids sur les arcs $c : E(G) \to \mathbb{R}_+$. Les sommets représentent les éléments de stockage, les arcs représentent des chemins à travers des logiques combinatoires, et les poids représentent la pire estimation du temps de transmission d'un signal. Un objectif important dans la conception des circuits intégrés (VLSI) de grande taille est de minimiser le temps du cycle de l'horloge T, c.-à-d. de trouver une fonction $a : V(G) \to \mathbb{R}$ telle que $a(v) + c((v, w)) \leq a(w) + T$ pour tout $(v, w) \in E(G)$, le nombre T étant aussi petit que possible. ($a(v)$ et $a(v) + T$ sont, respectivement, les «temps de départ» et les «temps d'arrivée au plus tard» d'un signal issu de v.)

   (a) Ramener la recherche du cycle d'horloge minimum T à un PROBLÈME DU CIRCUIT MOYEN MINIMUM.

   (b) Montrer qu'on peut calculer de manière efficace les nombres $a(v)$.

   (c) Montrer comment résoudre le problème quand certaines des quantités $a(v)$ sont fixées à l'avance.

   (Albrecht *et al.* [2002])

Références

Littérature générale :

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. [1993] : Network Flows. Prentice-Hall, Englewood Cliffs 1993, Chapters 4 and 5

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. [2001] : Introduction to Algorithms. Second Edition. MIT Press, Cambridge 2001, Chapters 24 and 25

Dreyfus, S.E. [1969] : An appraisal of some shortest path algorithms. Operations Research 17 (1969), 395–412

Gallo, G., Pallottino, S. [1988] : Shortest paths algorithms. Annals of Operations Research 13 (1988), 3–79

Gondran, M., Minoux, M. [1984] : Graphs and Algorithms. Wiley, Chichester 1984, Chapter 2

Lawler, E.L. [1976] : Combinatorial Optimization : Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Chapter 3

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 6–8

Tarjan, R.E. [1983] : Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Chapter 7

Références citées :

Ahuja, R.K., Mehlhorn, K., Orlin, J.B., Tarjan, R.E. [1990] : Faster algorithms for the shortest path problem. Journal of the ACM 37 (1990), 213–223

Albrecht, C., Korte, B., Schietke, J., Vygen, J. [2002] : Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip. Discrete Applied Mathematics 123 (2002), 103–127

Bazaraa, M.S., Langley, R.W. [1974] : A dual shortest path algorithm. SIAM Journal on Applied Mathematics 26 (1974), 496–501

Bellman, R.E. [1958] : On a routing problem. Quarterly of Applied Mathematics 16 (1958), 87–90

Chan, T.M. [2007] : More algorithms for all-pairs shortest paths in weighted graphs. Proceedings of the 39th Annual ACM Symposium on Theory of Computing (2007), 590–598

Cherkassky, B.V., Goldberg, A.V. [1999] : Negative-cycle detection algorithms. Mathematical Programming A 85 (1999), 277–311

Dial, R.B. [1969] : Algorithm 360 : shortest path forest with topological order. Communications of the ACM 12 (1969), 632–633

Dijkstra, E.W. [1959] : A note on two problems in connexion with graphs. Numerische Mathematik 1 (1959), 269–271

Edmonds, J., Karp, R.M. [1972] : Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM 19 (1972), 248–264

Fakcharoenphol, J., Rao, S. [2006] : Planar graphs, negative weight edges, shortest paths, and near linear time. Journal of Computer and System Sciences 72 (2006), 868–889

Floyd, R.W. [1962] : Algorithm 97 – shortest path. Communications of the ACM 5 (1962), 345

Ford, L.R. [1956] : Network flow theory. Paper P-923, The Rand Corporation, Santa Monica 1956

Fredman, M.L., Tarjan, R.E. [1987] : Fibonacci heaps and their uses in improved network optimization problems. Journal of the ACM 34 (1987), 596–615

Fredman, M.L., Willard, D.E. [1994] : Trans-dichotomous algorithms for minimum spanning trees and shortest paths. Journal of Computer and System Sciences 48 (1994), 533–551

Goldberg, A.V. [1995] : Scaling algorithms for the shortest paths problem. SIAM Journal on Computing 24 (1995), 494–504

Henzinger, M.R., Klein, P., Rao, S., Subramanian, S. [1997] : Faster shortest-path algorithms for planar graphs. Journal of Computer and System Sciences 55 (1997), 3–23

Johnson, D.B. [1977] : Efficient algorithms for shortest paths in sparse networks. Journal of the ACM 24 (1977), 1–13

Johnson, D.B. [1982] : A priority queue in which initialization and queue operations take $O(\log \log D)$ time. Mathematical Systems Theory 15 (1982), 295–309

Karp, R.M. [1978] : A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23 (1978), 309–311

Megiddo, N. [1979] : Combinatorial optimization with rational objective functions. Mathematics of Operations Research 4 (1979), 414–424

Megiddo, N. [1983] : Applying parallel computation algorithms in the design of serial algorithms. Journal of the ACM 30 (1983), 852–865

Moore, E.F. [1959] : The shortest path through a maze. Proceedings of the International Symposium on the Theory of Switching, Part II, Harvard University Press, 1959, 285–292

Orlin, J.B. [1993] : A faster strongly polynomial minimum cost flow algorithm. Operations Research 41 (1993), 338–350

Pettie, S. [2004] : A new approach to all-pairs shortest paths on real-weighted graphs. Theoretical Computer Science 312 (2004), 47–74

Pettie, S., Ramachandran, V. [2005] : Computing shortest paths with comparisons and additions. SIAM Journal on Computing 34 (2005), 1398–1431

Radzik, T. [1993] : Parametric flows, weighted means of cuts, and fractional combinatorial optimization. In : Complexity in Numerical Optimization (P.M. Pardalos, ed.), World Scientific, Singapore 1993

Raman, R. [1997] : Recent results on the single-source shortest paths problem. ACM SIGACT News 28 (1997), 81–87

Thorup, M. [1999] : Undirected single-source shortest paths with positive integer weights in linear time. Journal of the ACM 46 (1999), 362–394

Thorup, M. [2000] : On RAM priority queues. SIAM Journal on Computing 30 (2000), 86–109

Thorup, M. [2004] : Integer priority queues with decrease key in constant time and the single source shortest paths problem. Journal of Computer and System Sciences 69 (2004), 330–353

Warshall, S. [1962] : A theorem on boolean matrices. Journal of the ACM 9 (1962), 11–12

Zwick, U. [2002] : All pairs shortest paths using bridging sets and rectangular matrix multiplication. Journal of the ACM 49 (2002), 289–317

Chapitre 8

Flots dans les réseaux

Dans ce chapitre et dans le suivant nous étudierons les flots dans les réseaux : soit un graphe orienté G avec des capacités $u : E(G) \to \mathbb{R}_+$ associées aux arcs et deux sommets particuliers, s (la **source**) et t (le **puits**). Le quadruplet (G, u, s, t) sera appelé un **réseau**.

Notre objectif sera de faire passer la plus grande quantité possible d'unités de flot de s à t. Une solution de ce problème sera appelée un flot maximum.

Définition 8.1. *Soit un graphe orienté G avec des capacités $u : E(G) \to \mathbb{R}_+$, un* **flot** *est une fonction $f : E(G) \to \mathbb{R}_+$ vérifiant $f(e) \leq u(e)$ pour tout $e \in E(G)$. f satisfait la* **règle de conservation du flot** *en $v \in V(G)$ si*

$$\mathrm{ex}_f(v) \;:=\; \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) \;=\; 0.$$

Un flot vérifiant cette règle pour chaque sommet de G est une **circulation**.

Soit un réseau (G, u, s, t) ; un **flot de s à t** *est un flot f vérifiant $\mathrm{ex}_f(s) < 0$ et $\mathrm{ex}_f(v) = 0$ pour $v \in V(G) \setminus \{s, t\}$. La* **valeur** *d'un flot f de s à t sera égale à :* $\text{valeur}\,(f) := -\,\mathrm{ex}_f(s)$.

Nous pouvons maintenant formuler le problème de base de ce chapitre :

PROBLÈME DU FLOT MAXIMUM	
Instance	Un réseau (G, u, s, t).
Tâche	Trouver un flot de s à t de valeur maximum.

Nous pourrons supposer que G est simple, quitte à agréger les arcs parallèles en un seul arc.

Ce problème a de nombreuses applications. Étudions, par exemple, le PROBLÈME D'AFFECTATION DES TÂCHES : étant donné n tâches, leur temps d'exécution $t_1, \ldots, t_n \in \mathbb{R}_+$ et un sous-ensemble d'employés $S_i \subseteq \{1, \ldots, m\}$ pouvant participer à l'exécution des tâches $i \in \{1, \ldots, n\}$, soit $x_{ij} \in \mathbb{R}_+$, $i = 1, \ldots, n$ et

$j \in S_i$ le temps pendant lequel l'employé j travaille sur la tâche i. Il s'agit de calculer ces quantités de telle sorte que toutes les tâches soient finies : $\sum_{j \in S_i} x_{ij} = t_i$, $i = 1, \ldots, n$, et que le temps total nécessaire à l'exécution de toutes les tâches, $T(x) := \max_{j=1}^{m} \sum_{i:j \in S_i} x_{ij}$ soit minimisé. Au lieu d'utiliser la PROGRAMMATION LINÉAIRE, nous allons rechercher ici un algorithme combinatoire.

Nous allons trouver $T(x)$ par recherche binaire. Il s'agit alors de savoir si, pour T fixé, le système suivant : $x_{ij} \in \mathbb{R}_+$, $\sum_{j \in S_i} x_{ij} = t_i$ et $\sum_{i:j \in S_i} x_{ij} \leq T$, $i = 1, \ldots, n$ est réalisable. Construisons un graphe orienté biparti et associons un sommet v_i à chaque tâche i, un sommet w_j à chaque employé j et un arc (v_i, w_j) si et seulement si $j \in S_i$. Introduisons ensuite deux nouveaux sommets s, t ainsi que les arcs (s, v_i) pour tout i et (w_j, t) pour tout j. Soit G ce graphe. Définissons les capacités $u : E(G) \to \mathbb{R}_+$ par $u((s, v_i)) := t_i$ et $u(e) := T$ pour les autres arcs. Les solutions réalisables x telles que $T(x) \leq T$ correspondent aux flots de s à t de valeur $\sum_{i=1}^{n} t_i$ dans (G, u). Ce sont les flots maximum.

Nous décrirons au paragraphe 8.1 un algorithme de base pour le PROBLÈME DU FLOT MAXIMUM ; une des conséquences de cet algorithme sera le théorème flot-max/coupe-min, un des résultats les plus importants de l'optimisation combinatoire. Nous montrerons également que, quand les capacités sont entières, il existe toujours un flot optimal entier. La combinaison de ces deux résultats conduira au théorème de Menger sur les chemins et chaînes disjointes (paragraphe 8.2).

Les paragraphes 8.3, 8.4 et 8.5 décrivent des algorithmes efficaces pour le PROBLÈME DU FLOT MAXIMUM. Nous étudierons ensuite le problème de la coupe minimum. Le paragraphe 8.6 décrit une manière élégante pour représenter la capacité minimum d'une coupe séparant t de s (qui est égale à la valeur maximum d'un flot de s à t) pour toutes les paires de sommets s, t. Le paragraphe 8.7 montre comment trouver de manière efficace l'arête-connexité, c.-à-d. la capacité minimum d'une coupe d'un graphe non orienté.

8.1 Théorème flot-max/coupe-min

La définition du PROBLÈME DU FLOT MAXIMUM suggère la formulation suivante par PL :

$$\begin{array}{lll}
\max & \displaystyle\sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e & \\
\text{s.c.} & \displaystyle\sum_{e \in \delta^-(v)} x_e = \sum_{e \in \delta^+(v)} x_e & (v \in V(G) \setminus \{s, t\}) \\
& x_e \leq u(e) & (e \in E(G)) \\
& x_e \geq 0 & (e \in E(G)).
\end{array}$$

Puisque ce PL est borné et puisque le flot nul $f \equiv 0$ est réalisable :

Proposition 8.2. *Le* PROBLÈME DU FLOT MAXIMUM *a toujours une solution optimale.* □

Notons que, par le théorème 4.18, il existe un algorithme polynomial de résolution de ce problème ; cette méthode n'est cependant pas appropriée et nous rechercherons des algorithmes combinatoires qui n'utilisent pas la PROGRAMMATION LINÉAIRE.

Rappelons qu'une coupe séparant t de s dans G est un ensemble d'arcs $\delta^+(X)$ tel que $s \in X$ et $t \in V(G) \setminus X$. La **capacité** d'une coupe séparant t de s est la somme des capacités de ses arcs. Par la suite, la coupe minimum séparant t de s dans (G, u) sera la coupe séparant t de s de capacité minimum dans G.

Lemme 8.3. *Soit $A \subseteq V(G)$ tel que $s \in A, t \notin A$, et soit f un flot de s à t ;*
(a) $\text{valeur}\,(f) = \sum_{e\in\delta^+(A)} f(e) - \sum_{e\in\delta^-(A)} f(e)$.
(b) $\text{valeur}\,(f) \leq \sum_{e\in\delta^+(A)} u(e)$.

Preuve. (a) : puisque la règle de conservation est vérifiée pour $v \in A \setminus \{s\}$,

$$\begin{aligned}\text{valeur}\,(f) &= \sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e)\\ &= \sum_{v\in A}\left(\sum_{e\in\delta^+(v)} f(e) - \sum_{e\in\delta^-(v)} f(e)\right)\\ &= \sum_{e\in\delta^+(A)} f(e) - \sum_{e\in\delta^-(A)} f(e).\end{aligned}$$

(b) se déduit de (a) parce que $0 \leq f(e) \leq u(e)$ pour tout $e \in E(G)$. □

La valeur d'un flot maximum ne peut donc excéder la capacité d'une coupe de capacité minimum. En réalité, il y a égalité et, pour montrer cela, introduisons la notion de chaîne augmentante qui réapparaîtra de nombreuses fois dans ce livre.

Définition 8.4. *Si G est un graphe orienté soit $\overleftrightarrow{G} := (V(G), E(G) \dot\cup \{\overleftarrow{e} : e \in E(G)\})$ où, à tout arc $e = (v, w) \in E(G)$, sera associé un nouvel arc $\overleftarrow{e}$ allant de w à v. $\overleftarrow{e}$ sera appelé* **arc inversé** *de e et vice versa. Notons que si $e = (v, w), e' = (w, v) \in E(G)$, alors $\overleftarrow{e}$ et e' sont des arcs parallèles de $\overleftrightarrow{G}$.*

Soit G un graphe orienté muni de capacités $u : E(G) \to \mathbb{R}_+$ et d'un flot f ; définissons les **capacités résiduelles** *$u_f : E(\overleftrightarrow{G}) \to \mathbb{R}_+$ par $u_f(e) := u(e) - f(e)$ et $u_f(\overleftarrow{e}) := f(e)$ pour tout $e \in E(G)$. Le* **graphe résiduel** *G_f sera le graphe $(V(G), \{e \in E(\overleftrightarrow{G}) : u_f(e) > 0\})$.*

Soit un flot f et soit P un chemin (ou circuit) de G_f ; **augmenter** *f le long de P par γ signifie : si $e \in E(P)$ et $e \in E(G)$ ajouter γ à $f(e)$, sinon – si $e \in E(P)$ et $e = \overleftarrow{e_0}$ pour $e_0 \in E(G)$ – retrancher γ à $f(e_0)$.*

Si (G, u, s, t) est un réseau et f est un flot de s à t, un **chemin *f*-augmentant** *est un chemin de s à t dans le graphe résiduel G_f.*

Avec cette notion, l'algorithme suivant de Ford et Fulkerson [1957] pour le PROBLÈME DU FLOT MAXIMUM est naturel. Restreignons-nous d'abord au cas des capacités entières.

ALGORITHME DE FORD-FULKERSON

Input Un réseau (G, u, s, t) et $u : E(G) \to \mathbb{Z}_+$.

Output Un flot f de s à t de valeur maximum.

① $f(e) = 0$ pour tout $e \in E(G)$.

② Trouver un chemin f-augmentant P. **If** aucun chemin n'existe **then stop**.

③ $\gamma := \min_{e \in E(P)} u_f(e)$. Augmenter f le long de P par γ et **go to** ②.

Les arcs pour lesquels le minimum est atteint en ③ sont parfois appelés les arcs seuils. Le choix de γ assure que f est encore un flot. Puisque P est un flot de s à t, la règle de conservation reste vérifiée pour tous les sommets sauf s et t.

Trouver un chemin augmentant est facile (il s'agit de trouver un chemin de s à t dans G_f), mais nécessite un certain soin. En effet, si nous autorisons des capacités irrationnelles, un mauvais choix des chemins augmentants peut faire que l'algorithme ne se termine jamais (exercice 2).

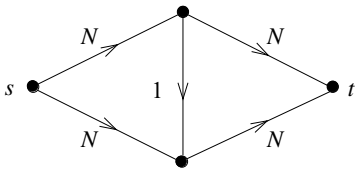


Figure 8.1.

Même dans le cas de capacités entières, on peut avoir un nombre exponentiel d'augmentations. Le réseau de la figure 8.1, où les nombres sont les capacités des arcs ($N \in \mathbb{N}$), illustre cela. Si à chaque itération on choisit un chemin augmentant de longueur 3, la valeur du flot s'accroît d'une unité ; le nombre total d'itérations est donc $2N$ alors que la taille de l'input est $O(\log N)$, les capacités étant codées sous forme binaire. Nous résoudrons ces difficultés au paragraphe 8.3.

Montrons que, quand l'algorithme se termine, f est un flot maximum :

Théorème 8.5. *Un flot f de s à t est maximum si et seulement s'il n'existe pas de chemin f-augmentant.*

Preuve. S'il existe un chemin augmentant P, ③ dans L'ALGORITHME DE FORD-FULKERSON calcule un flot de valeur plus grande et f n'est pas maximum. S'il

n'existe pas de chemin augmentant, t n'est pas connecté à s dans G_f. Soit R l'ensemble des sommets connectés à s dans G_f. La définition de G_f implique que $f(e) = u(e)$ si $e \in \delta_G^+(R)$ et $f(e) = 0$ si $e \in \delta_G^-(R)$.

Par le lemme 8.3 (a),

$$\text{valeur}\,(f) \;=\; \sum_{e \in \delta_G^+(R)} u(e)$$

et f est un flot maximum par le lemme 8.3 (b). □

Pour tout flot de s à t maximum on trouve donc une coupe séparant t de s de capacité égale à la valeur du flot. Grâce au lemme 8.3 (b), on obtient le théorème flot-max/coupe-min, résultat central en théorie des flots :

Théorème 8.6. (Ford et Fulkerson [1956], Dantzig et Fulkerson [1956]) *Dans un réseau, la valeur maximum d'un flot de s à t est égale à la capacité minimum d'une coupe séparant t de s.* □

Une preuve alternative a été proposée par Elias, Feinstein et Shannon [1956]. Le théorème flot-max/coupe-min peut aussi se déduire de la dualité en PL ; voir l'exercice 8 du chapitre 3.

Si les capacités sont entières, alors γ dans ③ de l'ALGORITHME DE FORD-FULKERSON est toujours entier. Puisqu'il existe un flot maximum de valeur finie (proposition 8.2), l'algorithme se termine après un nombre fini d'itérations. Nous avons donc l'importante conséquence suivante :

Corollaire 8.7. (Dantzig et Fulkerson [1956]) *Si les capacités du réseau sont entières, il existe un flot maximum entier.* □

Ce corollaire – appelé quelquefois le théorème du flot entier – peut également se démontrer en utilisant la totale unimodularité de la matrice d'incidence d'un graphe orienté (voir exercice 3).

Nous terminons ce paragraphe par une observation simple mais utile, le théorème de décomposition des flots :

Théorème 8.8. (Gallai [1958], Ford et Fulkerson [1962]) *Soit (G, u, s, t) un réseau et soit f un flot de s à t de G. Il existe une famille $\mathcal{P}$ de chemins de s à t et une famille $\mathcal{C}$ de circuits de G auxquels sont affectés des poids $w : \mathcal{P} \cup \mathcal{C} \to \mathbb{R}_+$ de telle sorte que $f(e) = \sum_{P \in \mathcal{P} \cup \mathcal{C}:\, e \in E(P)} w(P)$ pour tout $e \in E(G)$, $\sum_{P \in \mathcal{P}} w(P) =$ valeur (f), et $|\mathcal{P}| + |\mathcal{C}| \leq |E(G)|$.*

De plus, si f est entier, les poids w peuvent être choisis entiers.

Preuve. Construisons $\mathcal{P}$, $\mathcal{C}$ et w par induction sur le nombre d'arcs portant un flot non nul. Soit $e = (v_0, w_0)$ un arc tel que $f(e) > 0$.Si $w_0 \neq s$ il existe un arc (w_0, w_1) ayant un flot non nul. Posons $i := 1$. Si $w_i \in \{t, v_0, w_0, \ldots, w_{i-1}\}$ nous stoppons. Sinon, il existe un arc (w_i, w_{i+1}) ayant un flot non nul ; posons $i := i + 1$ et continuons. Ce processus s'arrête après au plus n étapes.

Faisons la même chose dans l'autre sens : si $v_0 \neq s$, il existe un arc (v_1, v_0) ayant un flot non nul, et ainsi de suite. Nous obtenons finalement soit un circuit, soit un chemin de s à t dans G dont tous les arcs portent un flot positif. Soit P ce circuit ou chemin. Soit $w(P) := \min_{e \in E(P)} f(e)$. Posons $f'(e) := f(e) - w(P)$ pour $e \in E(P)$ et $f'(e) := f(e)$ pour $e \notin E(P)$. Une application de l'hypothèse d'induction à f' termine la démonstration. □

8.2 Théorème de Menger

Quand toutes les capacités valent 1, les flots de s à t, par le corollaire 8.7 et le théorème 8.8 peuvent être identifiés à des collections de chemins de s à t et de circuits arc-disjoints. On obtient alors l'important théorème suivant :

Théorème 8.9. (Menger [1927]) *Soit G un graphe orienté (resp. non orienté) ; soient deux sommets s et t et soit $k \in \mathbb{N}$. Il existe k chemins de s à t arc-disjoints (resp. k chaînes de s à t arête-disjointes) si et seulement si t continue à être connecté à s après suppression de $k-1$ arcs (resp. arêtes) quelconques.*

Preuve. La nécessité est évidente. Montrons que la condition est suffisante dans le cas orienté : soit (G, u, s, t) un réseau ayant des capacités $u \equiv 1$ et tel que t soit connecté à s après suppression de $k-1$ arcs quelconques. La capacité minimum d'une coupe séparant t de s est au moins k. Par le théorème flot-max/coupe-min 8.6 et le corollaire 8.7, il existe un flot de s à t de valeur au moins k. Par le théorème 8.8 ce flot peut se décomposer en flots entiers portés par des chemins de s à t (et éventuellement par des circuits). Puisque toutes les capacités valent 1, il existe au moins k chemins de s à t arc-disjoints.

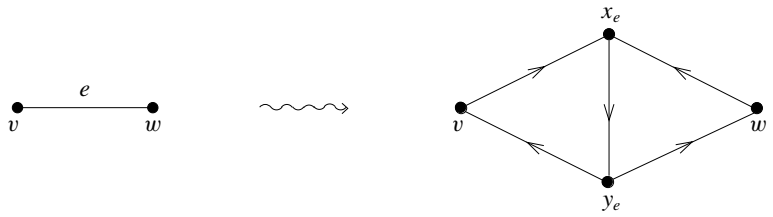


Figure 8.2.

Soit G un graphe non orienté ayant deux sommets s et t connectés entre eux après suppression de $k-1$ arêtes quelconques. Cette propriété reste vraie si on remplace chaque arête $e = (v, w)$ par cinq arcs (v, x_e), (w, x_e), (x_e, y_e), (y_e, v), (y_e, w), x_e, y_e étant de nouveaux sommets (voir figure 8.2). On a alors un graphe orienté G' et, par la première partie, k chemins de s à t arc-disjoints de G' qui se transforment facilement en k chaînes de s à t arête-disjointes de G. □

On peut facilement déduire le théorème flot-max/coupe-min (au moins quand les capacités sont rationnelles) du théorème de Menger. Étudions maintenant la version sommet-disjoints du théorème de Menger. Nous dirons que deux chemins (resp. chaînes) sont **sommet-disjoints** (resp. **sommet-disjointes**) s'ils n'ont (resp. si elles n'ont) aucun sommet interne (c.-à-d. différent d'une extrémité) commun.

Théorème 8.10. (Menger [1927]) *Soit G un graphe orienté (resp. non orienté) avec deux sommets non adjacents s et t et soit $k \in \mathbb{N}$. Il existe k chemins de s à t sommet-disjoints (resp. k chaînes de s à t sommet-disjointes) si et seulement si t reste connecté à s après suppression de $k-1$ sommets (différents de s et t) quelconques.*

Preuve. La condition est clairement nécessaire. On peut voir qu'elle est suffisante dans le cas orienté par le théorème 8.9, grâce à la construction suivante : remplaçons chaque sommet v de G par deux sommets v', v'' et un arc (v', v''). Chaque arc (v, w) de G est remplacé par (v'', w'). Si t' n'est plus connecté à s" après suppression de $k-1$ arcs du nouveau graphe G', cela implique que t n'est plus connecté à s après suppression d'un ensemble de $k-1$ sommets de G. De plus, les chemins de s'' à t' dans le nouveau graphe correspondent à des chemins de s à t dans l'ancien.

La version non orientée se déduit de la version orientée par la même construction que dans la preuve du théorème 8.9 (figure 8.2). □

Le corollaire suivant est une importante conséquence du théorème de Menger :

Corollaire 8.11. (Whitney [1932]) *Un graphe non orienté G ayant au moins deux sommets est k-arête-connexe si et seulement si pour chaque paire $s, t \in V(G)$ avec $s \neq t$ il existe k chaînes de s à t arête-disjointes.*

Un graphe non orienté G ayant au moins $k+1$ sommets est k-connexe si et seulement si pour chaque paire $s, t \in V(G)$ avec $s \neq t$ il existe k chaînes de s à t sommet-disjointes.

Preuve. La première condition découle du théorème 8.9.

Pour montrer la deuxième condition, soit G un graphe non orienté avec au moins $k+1$ sommets. Si la suppression de $k-1$ sommets déconnecte G, il ne peut exister k chaînes de s à t sommet-disjointes pour chaque paire $s, t \in V(G)$.

Inversement, s'il existe une paire $s, t \in V(G)$ n'étant pas connectée par k chaînes arête-disjointes, alors considérons deux cas. Si s et t sont non adjacents, G a $k-1$ sommets dont la suppression sépare s et t, par le théorème 8.10.

Si s et t sont adjacents et reliés par F arêtes parallèles, s et t ne sont pas connectés par $k - |F|$ chaînes arête-disjointes dans $G - F$; par le théorème 8.10, il existe un ensemble X de $k - |F| - 1$ sommets dont la suppression sépare s et t dans $G - F$. Soit $v \in V(G) \setminus (X \cup \{s, t\})$. Alors v n'est pas connecté à s ou t dans $(G - F) - X$; supposons que v ne soit pas connecté à s. v et s sont alors dans des composantes distinctes de $G - (X \cup \{t\})$. □

Dans de nombreuses applications on s'intéresse à la recherche de chemins arc/sommet disjoints, de chaînes arête/sommet disjointes entre plusieurs paires de

sommets. Aux quatre versions du théorème de Menger sont associées quatre versions du PROBLÈME DES CHEMINS DISJOINTS OU CHAÎNES DISJOINTES.

PROBLÈME DES CHEMINS ARC/SOMMET DISJOINTS, DES CHAÎNES ARÊTE/SOMMET DISJOINTES	
Instance	Deux graphes orientés ou non (G, H) ayant les mêmes sommets.
Tâche	Trouver une famille $(P_f)_{f \in E(H)}$ de chaînes arête/sommet disjointes ou chemins arc/sommet disjoints dans G tels que pour tout $f = (t, s)$ de H, P_f soit une chaîne ou un chemin de s à t.

Une telle famille est appelée **solution** de (G, H). Nous dirons que P_f **réalise** f. Les arcs ou arêtes de G seront les **offres**, les arcs ou arêtes de H seront les **demandes**. Un sommet incident à une demande sera un **sommet terminal**.

Le théorème de Menger correspond au cas où H est un ensemble de k arcs ou arêtes parallèles. Le PROBLÈME DES CHEMINS DISJOINTS OU CHAÎNES DISJOINTES sera étudié au chapitre 19. Notons ici le cas particulier du théorème de Menger :

Proposition 8.12. *Soit (G, H) une instance du* PROBLÈME DES CHEMINS ARC-DISJOINTS *où H est un ensemble d'arcs parallèles et $G + H$ est eulérien. Alors (G, H) a une solution.*

Preuve. Puisque $G + H$ est eulérien chaque arc, en particulier chaque arc $f \in E(H)$, appartient à un circuit C. Choisissons $C - f$ comme premier chemin ; supprimons C. Nous concluons par induction. □

8.3 Algorithme d'Edmonds-Karp

Comme l'exercice 2 le suggère, il est nécessaire de préciser l'étape ② de l'ALGORITHME DE FORD-FULKERSON. Au lieu de choisir un chemin augmentant quelconque, une bonne stratégie consiste à prendre le chemin augmentant ayant le nombre minimum d'arcs. Edmonds et Karp [1972] ont réussi à obtenir, grâce à cette idée, le premier algorithme polynomial pour le PROBLÈME DU FLOT MAXIMUM.

ALGORITHME D'EDMONDS-KARP	
Input	Un réseau (G, u, s, t).
Output	Un flot f de s à t de valeur maximum.

① $f(e) = 0$ pour tout $e \in E(G)$.

② Trouver un plus court chemin f-augmentant P. **If** il n'existe pas de chemin, **then stop**.

③ Calculer $\gamma := \min_{e \in E(P)} u_f(e)$. Augmenter f le long de P par γ et **go to** ②.

Cela signifie que ② de L'ALGORITHME DE FORD-FULKERSON doit être implémenté par BFS (voir paragraphe 2.3).

Lemme 8.13. *Soient $f_1, f_2, \ldots$ une suite de flots telle que f_{i+1} se déduit de f_i par augmentation le long de P_i, P_i étant un plus court chemin f_i-augmentant.*

(a) $|E(P_k)| \leq |E(P_{k+1})|$ *pour tout k.*

(b) $|E(P_k)| + 2 \leq |E(P_l)|$ *pour tout $k < l$ tel que $P_k \cup P_l$ contienne une paire d'arcs inversés.*

Preuve. (a) : soit G_1 le graphe résultant de $P_k \dot\cup P_{k+1}$ par suppression des arcs inversés. (Les arcs qui sont dans P_k et dans P_{k+1} apparaissent deux fois dans G_1.) Notons que $E(G_1) \subseteq E(G_{f_k})$, puisque tout arc dans $E(G_{f_{k+1}}) \setminus E(G_{f_k})$ est un arc inversé d'un arc de P_k.

Définissons H_1 comme étant deux copies de (t, s). $G_1 + H_1$ est manifestement eulérien. Donc, par la proposition 8.12, il existe deux chemins de s à t arc-disjoints Q_1 et Q_2. Puisque $E(G_1) \subseteq E(G_{f_k})$, Q_1 et Q_2 sont tous deux des chemins f_k-augmentants. Puisque P_k était un chemin f_k-augmentant plus court, $|E(P_k)| \leq |E(Q_1)|$ et $|E(P_k)| \leq |E(Q_2)|$. Donc

$$2|E(P_k)| \;\leq\; |E(Q_1)| + |E(Q_2)| \;\leq\; |E(G_1)| \;\leq\; |E(P_k)| + |E(P_{k+1})|,$$

ce qui implique $|E(P_k)| \leq |E(P_{k+1})|$.

(b) : par la partie (a), il suffit de montrer le résultat pour les indices k, l tels que $P_i \cup P_l$ ne contienne pas de paires d'arcs inversés pour $k < i < l$.

Comme précédemment, soit G_1 le graphe résultant de $P_k \dot\cup P_l$ par suppression des arcs inversés. Montrons que $E(G_1) \subseteq E(G_{f_k})$: en effet, $E(P_k) \subseteq E(G_{f_k})$, $E(P_l) \subseteq E(G_{f_l})$, et tout arc de $E(G_{f_l}) \setminus E(G_{f_k})$ doit être l'arc inversé d'un arc qui est sur un des chemins $P_k, P_{k+1}, \ldots, P_{l-1}$. Mais – par le choix de k et l – seul P_k parmi ces chemins contient l'arc inversé d'un arc de P_l.

Le graphe H_1 comme précédemment consiste en deux copies de (t, s). Puisque $G_1 + H_1$ est eulérien, il existe, par la proposition 8.12, deux chemins de s à t arc-disjoints Q_1 et Q_2. Q_1 et Q_2 sont f_k-augmentants. Puisque P_k est le plus court chemin f_k-augmentant, $|E(P_k)| \leq |E(Q_1)|$ et $|E(P_k)| \leq |E(Q_2)|$. Donc

$$2|E(P_k)| \;\leq\; |E(Q_1)| + |E(Q_2)| \;\leq\; |E(P_k)| + |E(P_l)| - 2$$

(puisque au moins deux arcs ont été supprimés). Cela termine la preuve. □

Théorème 8.14. (Edmonds et Karp [1972]) *Quelles que soient les capacités des arcs, l'*ALGORITHME D'EDMONDS-KARP *se termine après au plus $\frac{mn}{2}$ augmentations, m et n étant respectivement le nombre d'arcs et de sommets.*

Preuve. Soient $P_1, P_2, \ldots$ les chemins augmentants détectés dans l'ALGORITHME D'EDMONDS-KARP. Par le choix de γ dans l'étape ③ de l'algorithme, chaque chemin augmentant contient au moins un arc seuil.

Pour tout e, soit $P_{i_1}, P_{i_2}, \ldots$ la sous-suite de chemins augmentants contenant e comme arc seuil. Entre P_{i_j} $P_{i_{j+1}}$ il est évident qu'il existe un chemin augmentant

P_k ($i_j < k < i_{j+1}$) contenant $\overleftarrow{e}$. Par le lemme 8.13 (b), $|E(P_{i_j})| + 4 \leq |E(P_k)| + 2 \leq |E(P_{i_{j+1}})|$ pour tout j. Si les extrémités de e sont distinctes de s et t, $3 \leq |E(P_{i_j})| \leq n-1$ pour tout j, et il y a au plus $\frac{n}{4}$ chemins augmentants contenant e comme arc seuil. Dans le cas contraire, au plus un des chemins augmentants contient e ou $\overleftarrow{e}$ comme arc seuil.

Puisque tout chemin augmentant contient au moins un arc de $\overleftrightarrow{G}$ comme arc seuil, il y a au plus $|E(\overleftrightarrow{G})|\frac{n}{4} = \frac{mn}{2}$ chemins augmentants. □

Corollaire 8.15. *L'*ALGORITHME D'EDMONDS-KARP *résout le* PROBLÈME DU FLOT MAXIMUM *en un temps* $O(m^2n)$.

Preuve. Par le théorème 8.14 il y a au plus $\frac{mn}{2}$ augmentations. Chaque augmentation utilise BFS et nécessite un temps $O(m)$. □

8.4 Flots bloquants et algorithme de Fujishige

Au moment où Edmonds et Karp proposaient un algorithme polynomial pour le PROBLÈME DU FLOT MAXIMUM, Dinic [1970] indépendamment trouvait un algorithme encore plus performant fondé sur la définition suivante :

Définition 8.16. *Soit un réseau (G, u, s, t) et soit un flot f de s à t. Le* **graphe niveau** *G_f^L de G_f est le graphe*

$$\left(V(G), \left\{(e = (x, y) \in E(G_f) : \mathrm{dist}_{G_f}(s, x) + 1 = \mathrm{dist}_{G_f}(s, y)\right\}\right).$$

Notons que le graphe niveau est sans circuit et peut être simplement obtenu par BFS en un temps $O(m)$.

Le lemme 8.13 (a) montre que la longueur du plus court chemin augmentant dans l'ALGORITHME D'EDMONDS-KARP est non décroissante. Appelons **phase** une suite de chemins augmentants de même longueur obtenus au cours de l'algorithme. Soit f le flot au début de la phase. Par la preuve du lemme 8.13 (b), tous les chemins augmentants de cette phase sont aussi des chemins augmentants de G_f. Donc tous ces chemins sont des chemins de s à t du graphe niveau de G_f.

Définition 8.17. *Soit (G, u, s, t) un réseau ; un flot de s à t f est dit* **bloquant** *si $(V(G), \{e \in E(G) : f(e) < u(e)\})$ ne contient aucun chemin de s à t.*

L'union des chemins augmentants dans une phase forme un flot bloquant dans G_f^L. Remarquons qu'un flot bloquant n'est pas nécessairement maximum. Les observations précédentes suggèrent le schéma algorithmique suivant :

ALGORITHME DE DINIC	
Input	Un réseau (G, u, s, t).
Output	Un flot f de s à t de valeur maximum.

① $f(e) = 0$ pour tout $e \in E(G)$.

② Construire le graphe niveau G_f^L de G_f.

③ Trouver un flot bloquant de s à t f' dans G_f^L. **If** $f' = 0$ **then stop**.

④ Ajouter f' à f et **go to** ②.

Puisque la longueur d'un plus court chemin augmentant s'accroît de phase en phase, l'ALGORITHME DE DINIC s'arrête après au plus $n - 1$ phases. Il reste à montrer comment trouver efficacement un flot bloquant dans un graphe sans circuit. Dinic a obtenu une borne $O(nm)$ pour chaque phase, ce qui n'est pas difficile à montrer (voir exercice 14).

Cette borne a été améliorée à $O(n^2)$ par Karzanov [1974] ; voir aussi Malhotra, Kumar et Maheshwari [1978]. Des améliorations importantes sont dues à Cherkassky [1977], Galil [1980], Galil et Namaad [1980], Shiloach [1978], Sleator [1980], et Sleator et Tarjan [1983]. Les deux dernières références décrivent un algorithme en $O(m \log n)$ pour trouver un flot bloquant dans un graphe sans circuit en utilisant une structure de données appelée arbres dynamiques. En utilisant ce dernier résultat comme sous-programme de l'ALGORITHME DE DINIC, on obtient un algorithme en $O(mn \log n)$ pour le PROBLÈME DU FLOT MAXIMUM. Cependant nous ne décrirons pas ici ces algorithmes (voir Tarjan [1983]), car un algorithme encore plus rapide sera l'objet du paragraphe suivant.

Nous terminerons ce paragraphe en décrivant l'algorithme faiblement polynomial de Fujishige [2003], principalement à cause de sa simplicité :

ALGORITHME DE FUJISHIGE

Input Un réseau (G, u, s, t) et $u : E(G) \to \mathbb{Z}_+$.

Output Un flot f de s à t de valeur maximum.

① $f(e) = 0$ pour tout $e \in E(G)$. $\alpha := \max\{u(e) : e \in E(G)\}$.

② $i := 1$, $v_1 := s$, $X := \emptyset$, et $b(v) := 0$ pour tout $v \in V(G)$.

③ **For** $e = (v_i, w) \in \delta^+_{G_f}(v_i)$ avec $w \notin \{v_1, \ldots, v_i\}$ **do** :
$b(w) := b(w) + u_f(e)$. **If** $b(w) \geq \alpha$ **then** $X := X \cup \{w\}$.

④ **If** $X = \emptyset$ **then** :
$\alpha := \lfloor \frac{\alpha}{2} \rfloor$. **If** $\alpha = 0$ **then stop else go to** ②.

⑤ $i := i + 1$. Choisir $v_i \in X$ et $X := X \setminus \{v_i\}$.
If $v_i \neq t$ **then go to** ③.

⑥ $\beta(t) := \alpha$ et $\beta(v) := 0$ pour tout $v \in V(G) \setminus \{t\}$.
While $i > 1$ **do** :
For $e = (p, v_i) \in \delta^-_{G_f}(v_i)$ avec $p \in \{v_1, \ldots, v_{i-1}\}$ **do** :
$\beta' := \min\{\beta(v_i), u_f(e)\}$.
Augmenter f le long de e par β'.
$\beta(v_i) := \beta(v_i) - \beta'$ et $\beta(p) := \beta(p) + \beta'$.
$i := i - 1$.
⑦ **Go to** ②.

Théorème 8.18. L'ALGORITHME DE FUJISHIGE *résout le* PROBLÈME DU FLOT MAXIMUM *pour des graphes simples orientés G ayant des capacités entières $u : E(G) \to \mathbb{Z}_+$, en un temps $O(mn \log u_{\max})$, où $n := |V(G)|$, $m := |E(G)|$ et $u_{\max} := \max\{u(e) : e \in E(G)\}$.*

Preuve. Nous appellerons itération une suite d'étapes se terminant par ④ ou ⑦. Dans ②–⑤, $v_1, \ldots, v_i$ sera un ordre de sommets qui vérifiera toujours $b(v_j) = u_f(E^+(\{v_1, \ldots, v_{j-1}\}, \{v_j\})) \geq \alpha$ pour $j = 2, \ldots, i$. Dans ⑥ le flot f est augmenté par l'invariant $\sum_{v \in V(G)} \beta(v) = \alpha$, et le nouveau flot est un flot de s à t dont la valeur s'accroît d'une quantité α.

Donc, après au plus $n - 1$ itérations, α diminuera pour la première fois. Quand α devient égal à $\alpha' = \lfloor \frac{\alpha}{2} \rfloor \geq \frac{\alpha}{3}$ dans ④, nous avons une coupe séparant t de s, $\delta^+_{G_f}(\{v_1, \ldots, v_i\})$ dans G_f de capacité plus petite que $\alpha(|V(G)| - i)$ puisque $b(v) = u_f(E^+(\{v_1, \ldots, v_i\}, \{v\})) < \alpha$ pour tout $v \in V(G) \setminus \{v_1, \ldots, v_i\}$. Par le lemme 8.3 (b), un flot de s à t maximum dans G_f a une valeur plus petite que $\alpha(n - i) < 3\alpha' n$. Donc après un nombre d'itérations inférieur à $3n$, α décroîtra de nouveau. Si α décroît de 1 à 0, nous obtenons une coupe séparant t de s de capacité 0 dans G_f ; donc f est maximum. Comme α décroît au plus $1 + \log u_{\max}$ fois avant d'atteindre la valeur 0, et comme chaque itération entre deux modifications de α prend un temps $O(m)$ le temps total de calcul est $O(mn \log u_{\max})$. □

Cette technique de réduction d'échelle est souvent utile et réapparaîtra au chapitre 9. Fujishige [2003] a proposé une variante sans réduction d'échelle, où v_i dans ⑤ est choisi comme un sommet atteignant le maximum de $\max\{b(v) : v \in V(G) \setminus \{v_1, \ldots, v_{i-1}\}\}$. L'ordre résultant, appelé ordre MA (maximum adjacence) sera de nouveau étudié au paragraphe 8.7. Le temps de calcul de cette variante est un peu plus élevé et cet algorithme n'est pas fortement polynomial (Shioura [2004]) ; voir l'exercice 17.

8.5 Algorithme de Goldberg-Tarjan

Nous décrirons dans ce paragraphe l'ALGORITHME PUSH-RELABEL de Goldberg et Tarjan [1988]. Nous en déduirons une borne $O(n^2\sqrt{m})$ du temps de calcul.

Des mises en œuvre sophistiquées utilisant les arbres dynamiques (voir Sleator et Tarjan [1983]) permettent d'obtenir des complexités $O\left(nm\ \log \frac{n^2}{m}\right)$ (Goldberg et Tarjan [1988]) et $O\left(nm\ \log\left(\frac{n}{m}\sqrt{\log u_{\max}}+2\right)\right)$, $u_{\max}$ étant la capacité (entière) maximum d'un arc (Ahuja, Orlin et Tarjan [1989]). Les meilleures bornes connues sont $O\left(nm \log_{2+m/(n\log n)} n\right)$ (King, Rao et Tarjan [1994]) et

$$O\left(\min\{m^{1/2}, n^{2/3}\} m \log\left(\frac{n^2}{m}\right) \log u_{\max}\right)$$

(Goldberg et Rao [1998]).

Par le théorème 8.5, f est un flot de s à t maximum si et seulement si :

- $\mathrm{ex}_f(v) = 0 \quad$ pour tout $v \in V(G) \setminus \{s,t\}$
- il n'existe pas de chemin f-augmentant.

Dans les algorithmes précédents la première condition est toujours satisfaite et l'algorithme se termine quand la deuxième est vérifiée. Dans l'ALGORITHME PUSH-RELABEL la seconde condition est satisfaite et l'algorithme s'arrête quand la première est vérifiée. Introduisons d'abord la notion de pré-flot de s à t.

Définition 8.19. *Soit un réseau (G,u,s,t) ; un* **pré-flot de *s* à *t*** *est une fonction $f : E(G) \to \mathbb{R}_+$ vérifiant $f(e) \le u(e)$ pour tout $e \in E(G)$ et $\mathrm{ex}_f(v) \ge 0$ pour tout $v \in V(G) \setminus \{s\}$. Un sommet $v \in V(G) \setminus \{s,t\}$ sera dit* **actif** *si $\mathrm{ex}_f(v) > 0$.*

Un pré-flot est un flot si et seulement s'il n'y a pas de sommets actifs.

Définition 8.20. *Soient (G,u,s,t) un réseau et f un pré-flot de s à t. Une* **distance étiquetée** *est une fonction $\psi : V(G) \to \mathbb{Z}_+$ telle que $\psi(t) = 0$, $\psi(s) = n$ et $\psi(v) \le \psi(w)+1$ pour tout $(v,w) \in E(G_f)$. Un arc $e = (v,w) \in E(\overleftrightarrow{G})$ est appelé* **admissible** *si $e \in E(G_f)$ et $\psi(v) = \psi(w)+1$.*

Si ψ est une distance étiquetée, $\psi(v)$ (pour $v \neq s$) est une borne inférieure de la distance de v à t (nombre d'arcs d'un plus court chemin de v à t) dans G_f.

L'ALGORITHME PUSH-RELABEL que nous allons décrire utilise un pré-flot f de s à t et une distance étiquetée ψ. Il commence avec un pré-flot égal à la capacité pour chaque arc d'origine s et égal à zéro sur les autres arcs. La distance étiquetée initiale est $\psi(s) = n$ et $\psi(v) = 0$ pour tout $v \in V(G) \setminus \{s\}$.

Puis l'algorithme effectue une suite d'OPÉRATIONS PUSH (actualisation de f) et d'OPÉRATIONS RELABEL (actualisation de ψ) dans un ordre quelconque.

ALGORITHME PUSH-RELABEL	
Input	Un réseau (G,u,s,t).
Output	Un flot f de s à t de valeur maximum.

① $f(e) := u(e)$ pour tout $e \in \delta^+(s)$.
$f(e) := 0$ pour tout $e \in E(G) \setminus \delta^+(s)$.

② $\psi(s) := n$ et $\psi(v) := 0$ pour tout $v \in V(G) \setminus \{s\}$.

③ **While** il existe un sommet actif **do** :
Soit v un sommet actif.
If aucun $e \in \delta^+_{G_f}(v)$ n'est admissible
then RELABEL(v),
else soit $e \in \delta^+_{G_f}(v)$ un arc admissible et PUSH(e).

ALGORITHME PUSH(e)

① $\gamma := \min\{\mathrm{ex}_f(v), u_f(e)\}$, où v est tel que $e \in \delta^+_{G_f}(v)$.

② Augmenter f le long de e par γ.

ALGORITHME RELABEL(v)

① $\psi(v) := \min\{\psi(w) + 1 : e = (v, w) \in E(G_f)\}$.

Proposition 8.21. *À chaque itération de l'*ALGORITHME PUSH-RELABEL *f est un pré-flot de s à t et ψ est une distance étiquetée par rapport à f.*

Preuve. Montrons que ces propriétés restent vérifiées après les procédures PUSH et RELABEL. Il est évident que, après une opération PUSH, f reste un pré-flot de s à t. Une opération RELABEL ne change pas f. De plus, ψ reste une distance étiquetée après une opération RELABEL.

Il reste à montrer qu'après une opération PUSH, ψ est une distance étiquetée à l'égard du nouveau pré-flot. Vérifions que $\psi(a) \leq \psi(b) + 1$ pour tous les nouveaux arcs (a, b) dans G_f. Mais quand nous appliquons PUSH(e) sur $e = (v, w)$, le seul nouvel arc possible de G_f est l'arc inversé de e, et nous avons alors $\psi(w) = \psi(v) - 1$, puisque e est admissible. □

Lemme 8.22. *Si f est un pré-flot de s à t et ψ est une distance étiquetée par rapport à f, alors :*

(a) *s est connecté à tout sommet actif v de G_f.*

(b) *Si w est connecté à un sommet v dans G_f, $\psi(v) \leq \psi(w) + n - 1$.*

(c) *t n'est pas connecté à s dans G_f.*

Preuve. (a) : soit v un sommet actif, et soit R l'ensemble des sommets connectés à v dans G_f. Alors $f(e) = 0$ pour tout $e \in \delta^-_G(R)$. Donc

$$\sum_{w \in R} \mathrm{ex}_f(w) \;=\; \sum_{e \in \delta^-_G(R)} f(e) - \sum_{e \in \delta^+_G(R)} f(e) \;\leq\; 0.$$

Mais v est actif et $\mathrm{ex}_f(v) > 0$; il existe donc un sommet $w \in R$ tel que $\mathrm{ex}_f(w) < 0$. Puisque f est un pré-flot de s à t, ce sommet est s.

(b) : supposons qu'il existe un chemin de v à w dans G_f, ayant pour sommets $v = v_0, v_1, \ldots, v_k = w$. Puisqu'il existe une distance étiquetée ψ par rapport à f, $\psi(v_i) \leq \psi(v_{i+1}) + 1$, $i = 0, \ldots, k-1$ et $\psi(v) \leq \psi(w) + k$. Notons que $k \leq n-1$.
(c) : se déduit de (b) puisque $\psi(s) = n$ et $\psi(t) = 0$. □

Grâce à (c) on peut déduire le résultat suivant :

Théorème 8.23. *Quand l'algorithme se termine, f est un flot de s à t maximum.*

Preuve. f est un flot de s à t puisqu'il n'y a pas de sommets actifs. Par le lemme 8.22(c), il n'existe pas de chemin augmentant. Par le théorème 8.5, f est maximum. □

Il faut maintenant calculer le nombre d'opérations PUSH et RELABEL.

Lemme 8.24.

(a) *$\psi(v)$ augmente strictement à chaque opération* RELABEL(v)*, et ne décroît jamais pour tout $v \in V(G)$.*

(b) *À chaque étape de l'algorithme, $\psi(v) \leq 2n-1$ pour tout $v \in V(G)$.*

(c) *Aucun sommet n'est relabelisé plus que $2n-1$ fois. Le nombre maximum d'*OPÉRATIONS RELABEL *est $2n^2 - n$.*

Preuve. (a) : ψ est modifié seulement par la procédure RELABEL. Si aucun arc $e \in \delta^+_{G_f}(v)$ n'est admissible, $\psi(v)$ augmente strictement à chaque opération RELABEL(v) parce que ψ est toujours une distance étiquetée.
(b) : $\psi(v)$ ne change que si v est actif. Par le lemme 8.22(a) et (b), $\psi(v) \leq \psi(s) + n - 1 = 2n - 1$.
(c) : se déduit directement de (a) et (b). □

Calculons le nombre d'OPÉRATIONS PUSH ; nous distinguerons les opérations PUSH **saturés** (où $u_f(e) = 0$ après le push) et les opérations PUSH **non saturés**.

Lemme 8.25. *Le nombre d'opérations push saturés est au plus $2mn$.*

Preuve. Après chaque push saturé de v à w, aucun autre push ne peut se produire jusqu'à ce que $\psi(w)$ augmente de 2 au moins, un push se produise de w à v et $\psi(v)$ augmente de 2. Cela montre, en utilisant le lemme 8.24(a) et (b), qu'il y a au plus n opérations push saturés sur chaque arc $(v, w) \in E(\overleftrightarrow{G})$. □

Le nombre d'opérations push non saturés est en général d'un ordre $n^2 m$ (voir exercice 19). En choisissant un sommet actif v avec $\psi(v)$ maximum dans ③ nous obtenons une meilleure borne. Nous pourrons supposer que $n \leq m \leq n^2$.

Lemme 8.26. *Si nous choisissons comme sommet dans ③ de l'*ALGORITHME PUSH-RELABEL *le sommet actif v tel que $\psi(v)$ soit maximum, le nombre d'opérations push non saturés est au plus $8n^2\sqrt{m}$.*

Preuve. Appelons phase le temps entre deux changements consécutifs de $\psi^* := \max\{\psi(v) : v \text{ actif}\}$. Comme ψ^* peut augmenter seulement en relabelisant, son nombre d'augmentations est au plus $2n^2$. Comme initialement $\psi^* = 0$, ψ^* peut diminuer au plus $2n^2$ fois, et le nombre total de phases est au plus $4n^2$.

Nous dirons qu'une phase est modeste si elle contient au plus $\sqrt{m}$ opérations push non saturés et qu'elle est importante dans le cas contraire. Clairement, il existe au plus $4n^2\sqrt{m}$ opérations push non saturés dans les phases modestes.

Soit

$$\Phi := \sum_{v \in V(G): v \text{ actif}} |\{w \in V(G) : \psi(w) \leq \psi(v)\}|.$$

Initialement $\Phi \leq n^2$. Une étape de relabelisation peut augmenter Φ par au plus n. Un push saturé peut augmenter Φ par au plus n. Un push non saturé n'augmente pas Φ. Puisque $\Phi = 0$ à la fin, la diminution totale de Φ est, au plus, $n^2 + n(2n^2 - n) + n(2mn) \leq 4mn^2$.

Considérons maintenant les opérations push non saturés pour les phases importantes. Chacune de ces opérations pousse un flot sur un arc (v, w) pour lequel $\psi(v) = \psi^* = \psi(w) + 1$ rend v inactif et rend éventuellement w actif.

Comme la phase se termine en relabelisant ou en rendant inactif le dernier sommet actif v tel que $\psi(v) = \psi^*$, l'ensemble des sommets w tels que $\psi(w) = \psi^*$ reste constant durant la phase, et contient plus que $\sqrt{m}$ sommets puisque la phase est importante. Donc chaque opération push non saturé dans une phase importante diminue Φ d'au moins $\sqrt{m}$. Donc le nombre total d'opérations push non saturés dans une phase importante est, au plus, $\frac{4mn^2}{\sqrt{m}} = 4n^2\sqrt{m}$. □

Cette preuve est due à Cheriyan et Mehlhorn [1999]. Finalement nous avons :

Théorème 8.27. (Goldberg et Tarjan [1988], Cheriyan et Maheshwari [1989], Tunçel [1994]) *L'*ALGORITHME PUSH-RELABEL *résout le* PROBLÈME DU FLOT MAXIMUM *et peut s'exécuter en un temps* $O(n^2\sqrt{m})$.

Preuve. L'algorithme répond correctement par la proposition 8.21 et le théorème 8.23.

Comme dans le lemme 8.26, choisissons comme sommet dans ③ le sommet actif v avec $\psi(v)$ maximum. Maintenons la trace de listes doublement chaînées $L_0, \ldots, L_{2n-1}$, où L_i contient les sommets actifs v tels que $\psi(v) = i$. Ces listes sont réactualisées à chaque opération PUSH et RELABEL en temps constant.

Nous pouvons commencer en décrivant L_i avec $i = 0$. Quand un sommet est relabelisé, nous augmentons i. Quand une liste L_i est vide pour l'indice courant i (après avoir rendu inactif le dernier sommet actif), nous diminuons i jusqu'à ce que L_i soit non vide. Comme nous augmentons i au plus $2n^2$ fois par le lemme 8.24(c), nous diminuons également i au plus $2n^2$ fois.

Comme seconde structure de données, nous gardons en mémoire une liste doublement chaînée A_v contenant les arcs admissibles sortant de v, pour chaque sommet v. Ceux-ci peuvent être mis à jour dans chaque opération PUSH en temps

constant, et dans chaque opération RELABEL en un temps proportionnel au nombre total d'arcs incidents au sommet qui est relabelisé.

Donc RELABEL(v) nécessite un temps $O(|\delta_G(v)|)$ et, par le lemme 8.24(c), le temps pour les opérations RELABEL est $O(mn)$. Chaque opération PUSH se fait en un temps constant et, par le lemme 8.25 et le lemme 8.26, le temps total des opérations PUSH est $O(n^2\sqrt{m})$. □

8.6 Arbres de Gomory-Hu

Tout algorithme pour le PROBLÈME DU FLOT MAXIMUM apporte également une solution au problème suivant :

PROBLÈME DE LA COUPE DE CAPACITÉ MINIMUM	
Instance	Un réseau (G, u, s, t).
Tâche	Une coupe séparant t de s de capacité minimum.

Proposition 8.28. *Le* PROBLÈME DE LA COUPE DE CAPACITÉ MINIMUM *se résout avec la même complexité que le* PROBLÈME DU FLOT MAXIMUM*, en particulier en un temps* $O(n^2\sqrt{m})$.

Preuve. Calculons un flot f de s à t maximum dans (G, u, s, t) ; soit X l'ensemble des sommets connectés à s dans G_f. On trouve X par l'ALGORITHME DE BALAYAGE DE GRAPHES en temps linéaire (proposition 2.17). Par le lemme 8.3 et le théorème 8.5, $\delta^+_G(X)$ est une coupe séparant t de s de capacité minimum. Le théorème 8.27 fournit une complexité $O(n^2\sqrt{m})$ (ce n'est pas la meilleure). □

Dans ce paragraphe, nous rechercherons les coupes séparant s et t de capacité minimum pour chaque paire de sommets s, t dans un graphe non orienté G muni de capacités $u : E(G) \to \mathbb{R}_+$.

Ce problème peut se ramener au précédent : pour toutes les paires $s, t \in V(G)$ nous résolvons le PROBLÈME DE LA COUPE DE CAPACITÉ MINIMUM dans (G', u', s, t), où (G', u') est obtenu à partir de (G, u) en remplaçant chaque arête e d'extrémités v, w par deux arcs de direction opposée (v, w) et (v, w) ayant la même capacité que e. De cette manière, nous obtenons toutes les coupes minimum séparant s et t pour toutes les paires s, t après $\binom{n}{2}$ calculs de flots.

Ce paragraphe est consacré à la méthode élégante de Gomory et Hu [1961], qui nécessite seulement $n - 1$ calculs de flots. Nous en verrons des applications aux paragraphes 12.3 et 20.3.

Définition 8.29. *Soit G un graphe non orienté ayant des capacités $u : E(G) \to \mathbb{R}_+$. Soient $s, t \in V(G)$; la capacité minimum d'une coupe séparant s et t sera appelée l'arête-connexité locale de s et t et sera notée λ_{st}.*

L'arête-connexité d'un graphe est, de manière évidente, le minimum de l'arête-connexité locale quand les capacités valent 1.

Lemme 8.30. *Soient $i, j, k \in V(G)$; alors $\lambda_{ik} \geq \min(\lambda_{ij}, \lambda_{jk})$.*

Preuve. Soit $\delta(A)$ une coupe avec $i \in A$, $k \in V(G) \setminus A$ et $u(\delta(A)) = \lambda_{ik}$. Si $j \in A$, $\delta(A)$ sépare j et k et $u(\delta(A)) \geq \lambda_{jk}$. Si $j \in V(G) \setminus A$, $\delta(A)$ sépare i et j et $u(\delta(A)) \geq \lambda_{ij}$. Donc, $\lambda_{ik} = u(\delta(A)) \geq \min(\lambda_{ij}, \lambda_{jk})$. □

Notons que cette condition est également suffisante : si des nombres $(\lambda_{ij})_{1 \leq i,j \leq n}$ avec $\lambda_{ij} = \lambda_{ji}$ vérifient cette condition, ils représentent l'arête-connexité locale d'un graphe (voir exercice 23).

Définition 8.31. *Soit G un graphe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$. Un* **arbre de Gomory-Hu** *associé à (G, u) est un arbre T tel que $V(T) = V(G)$ et*

$$\lambda_{st} = \min_{e \in E(P_{st})} u(\delta_G(C_e)) \quad \textit{pour } s, t \in V(G),$$

P_{st} étant l'unique chaîne de s à t dans T et C_e et $V(G) \setminus C_e$ étant les composantes connexes de $T - e$ pour $e \in E(T)$.

Nous verrons que tout graphe non orienté G possède un arbre de Gomory-Hu. Cela implique que, pour toute paire $s, t \in V(G)$, il existe une coupe minimum séparant s et t qui appartient à une liste fixée de $n - 1$ coupes.

En général, un arbre de Gomory-Hu n'est pas un sous-graphe de G. Par exemple, soit $G = K_{3,3}$ et $u \equiv 1$; $\lambda_{st} = 3$ pour toute paire $s, t \in V(G)$. Il est facile de voir que les arbres de Gomory-Hu de (G, u) sont les étoiles à cinq branches.

L'idée principale pour construire un arbre de Gomory-Hu est la suivante. Tout d'abord choisissons $s, t \in V(G)$ et cherchons une coupe minimum $\delta(A)$ séparant s et t. Soit $B := V(G) \setminus A$. Contractons ensuite A (resp. B) en un sommet, choisissons $s', t' \in B$ (resp. $s', t' \in A$) et cherchons une coupe minimum séparant s' et t' dans le graphe contracté G'. Poursuivons ce processus en choisissant toujours, une paire s', t' de sommets non encore séparés par une des coupes précédentes. À chaque étape, contractons – si $E(A', B')$ est la coupe actuelle – A' ou B', en fonction de la partie qui ne contient pas s' et t'.

Finalement toute paire de sommets sera séparée, et nous aurons alors un total de $n - 1$ coupes. L'observation capitale est qu'une coupe minimum séparant s' et t' dans le graphe contracté G' est aussi une coupe minimum séparant s' et t' dans G. Cela est l'objet du lemme suivant. Notons que quand un ensemble A de sommets dans (G, u) est contracté, la capacité de chaque arête de G' est la capacité de l'arête correspondante dans G.

Lemme 8.32. *Soit G un graphe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$. Soient $s, t \in V(G)$ et $\delta(A)$ une coupe minimum séparant s et t dans (G, u). Soient $s', t' \in V(G) \setminus A$, et supposons que (G', u') provienne de (G, u) en contractant A en un sommet. Soit $\delta(K \cup \{A\})$ une coupe minimum séparant s' et t' dans (G', u') ; alors $\delta(K \cup A)$ est une coupe minimum séparant s' et t' dans (G, u).*

Preuve. Soient s, t, A, s', t', G', u' comme dans l'énoncé. On peut toujours supposer que $s \in A$. Il suffit alors de montrer qu'il existe une coupe minimum $\delta(A')$

séparant s' et t' dans (G, u) telle que $A \subset A'$. Soit donc $\delta(C)$ une coupe minimum séparant s' et t' dans (G, u). On peut supposer que $s \in C$.

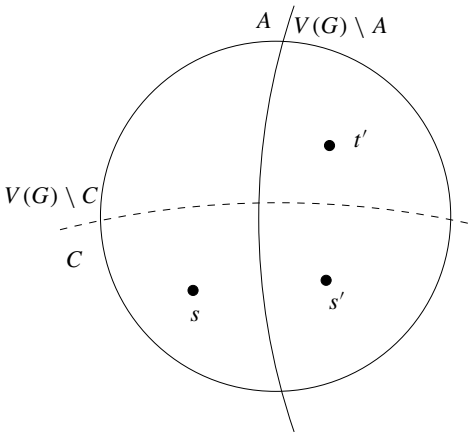


Figure 8.3.

Puisque $u(\delta(\cdot))$ est sous-modulaire (voir le lemme 2.1(c)), $u(\delta(A))+u(\delta(C)) \geq u(\delta(A\cap C))+u(\delta(A\cup C))$. $\delta(A\cap C)$ est une coupe séparant s et t et $u(\delta(A\cap C)) \geq \lambda_{st} = u(\delta(A))$. Donc $u(\delta(A\cup C)) \leq u(\delta(C)) = \lambda_{s't'}$, ce qui prouve que $\delta(A\cup C)$ est une coupe minimum séparant s' et t' (voir figure 8.3). □

Nous présentons maintenant un algorithme qui construit un arbre de Gomory-Hu. Les sommets des arbres intermédiaires T seront des sous-ensembles de sommets du graphe original qui forment une partition de $V(G)$. Le seul sommet de T est, au départ, $V(G)$. À chaque itération, un sommet de T contenant au moins deux sommets de G est choisi et divisé en deux sommets.

ALGORITHME DE GOMORY-HU

Input Un graphe non orienté G et des capacités $u : E(G) \to \mathbb{R}_+$.

Output Un arbre de Gomory-Hu T pour (G, u).

① $V(T) := \{V(G)\}$ et $E(T) := \emptyset$.

② Choisir $X \in V(T)$ avec $|X| \geq 2$. **If** X n'existe pas **then go to** ⑥.

③ Choisir $s, t \in X$ avec $s \neq t$.
For chaque composante connexe C de $T - X$ **do** : $S_C := \bigcup_{Y\in V(C)} Y$.
Soit (G', u') provenant de (G, u) en contractant S_C en un sommet v_C pour chaque composante connexe C de $T - X$.
(Donc $V(G') = X \cup \{v_C : C$ est une composante connexe de $T - X\}$.)

④ Trouver une coupe minimum $\delta(A')$ séparant s et t dans (G', u'). Soit $B' := V(G') \setminus A'$.
$A := \left(\bigcup_{v_C \in A' \setminus X} S_C \right) \cup (A' \cap X)$ et $B := \left(\bigcup_{v_C \in B' \setminus X} S_C \right) \cup (B' \cap X)$.

⑤ $V(T) := (V(T) \setminus \{X\}) \cup \{A \cap X, B \cap X\}$.
For chaque arête $e = (X, Y) \in E(T)$ incidente au sommet X **do** :
If $Y \subseteq A$ **then** $e' := (A \cap X, Y)$ **else** $e' := (B \cap X, Y)$.
$E(T) := (E(T) \setminus \{e\}) \cup \{e'\}$ et $w(e') := w(e)$.
$E(T) := E(T) \cup \{(A \cap X, B \cap X)\}$.
$w((A \cap X, B \cap X)) := u'(\delta_{G'}(A'))$.
Go to ②.

⑥ Remplacer tout $\{x\} \in V(T)$ par x et tout $(\{x\}, \{y\}) \in E(T)$ par (x, y).
Stop.

La figure 8.4 illustre la modification de T dans ⑤. Pour montrer que l'algorithme répond correctement, montrons d'abord le lemme suivant :

Lemme 8.33. *À chaque fin de l'étape ④ nous avons :*

(a) $A \dot\cup B = V(G)$.

(b) $E(A, B)$ *est une coupe minimum séparant* s *et* t *dans* (G, u).

Preuve. Les éléments de $V(T)$ sont toujours des sous-ensembles non vides de $V(G)$; $V(T)$ est bien une partition de $V(G)$, ce qui prouve (a).

Montrons maintenant (b). Cette propriété est trivialement vraie à la première itération (puisqu'on a $G' = G$). Montrons qu'elle est vérifiée à chaque itération.

Soient $C_1, \ldots, C_k$ les composantes connexes de $T - X$. Contractons-les une par une ; pour $i = 0, \ldots, k$ soit (G_i, u_i) provenant de (G, u) en contractant chaque $S_{C_1}, \ldots, S_{C_i}$ en un seul sommet. (G_k, u_k) est le graphe qui est nommé (G', u') en ③ de l'algorithme.

Observation : pour toute coupe minimum $\delta(A_i)$ séparant s et t dans (G_i, u_i), $\delta(A_{i-1})$ est une coupe séparant s et t dans (G_{i-1}, u_{i-1}), où

$$A_{i-1} := \begin{cases} (A_i \setminus \{v_{C_i}\}) \cup S_{C_i} & \text{si } v_{C_i} \in A_i \\ A_i & \text{si } v_{C_i} \notin A_i \end{cases}.$$

En appliquant cette observation pour $k, k-1, \ldots, 1$ nous obtenons (b).

Pour montrer l'observation, soit $\delta(A_i)$ une coupe minimum séparant s et t dans (G_i, u_i). Par notre hypothèse que (b) est vrai pendant les itérations précédentes, $\delta(S_{C_i})$ est une coupe minimum séparant s_i et t_i dans (G, u) pour une paire appropriée $s_i, t_i \in V(G)$. De plus, $s, t \in V(G) \setminus S_{C_i}$. Il suffit d'appliquer le lemme 8.32 pour terminer la preuve. □

Lemme 8.34. *À toute étape de l'algorithme (jusqu'à ce que ⑥ soit atteint), pour tout* $e \in E(T)$

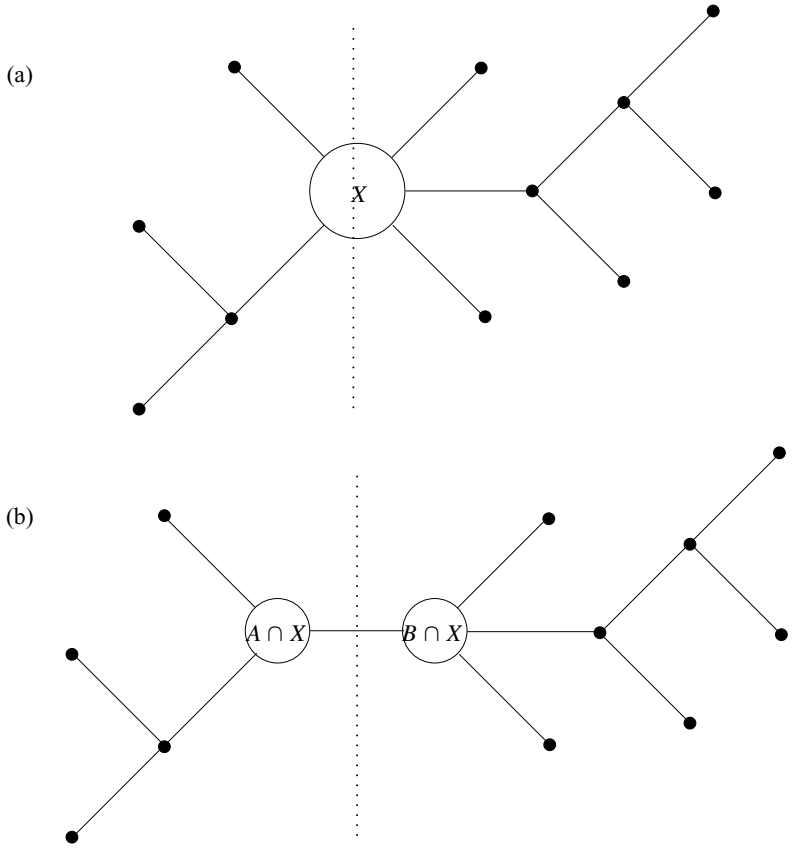


Figure 8.4.

$$w(e) \;=\; u\left(\delta_G\left(\bigcup_{Z\in C_e} Z\right)\right),$$

où C_e et $V(T) \setminus C_e$ sont les composantes connexes de $T - e$. De plus, pour tout $e = (P, Q) \in E(T)$ il existe des sommets $p \in P$ et $q \in Q$ avec $\lambda_{pq} = w(e)$.

Preuve. Les deux affirmations sont trivialement vraies au début de l'algorithme quand T ne contient pas d'arêtes ; montrons qu'elles restent toujours vraies. Soit X le sommet de T choisi en ② à une itération de l'algorithme. Soient s, t, A', B', A, B comme indiqué en ③ et ensuite ④. Nous pouvons toujours supposer que $s \in A'$.

Les arêtes de T non incidentes à X ne sont pas affectées par ⑤. Pour la nouvelle arête $(A \cap X, B \cap X)$, $w(e)$ est correctement défini et nous avons $\lambda_{st} = w(e)$, $s \in A \cap X$, $t \in B \cap X$.

Soit $e = (X, Y)$ une arête remplacée par e' dans ⑤. Nous pouvons supposer que $Y \subseteq A$; donc $e' = (A \cap X, Y)$. Si les deux affirmations sont vraies pour e, nous

dirons qu'elles restent vraies pour e'. C'est évident pour la première affirmation puisque $w(e) = w(e')$ et $u\left(\delta_G\left(\bigcup_{Z\in C_e} Z\right)\right)$ ne change pas.

Pour montrer la seconde affirmation, supposons qu'il existe $p \in X, q \in Y$ avec $\lambda_{pq} = w(e)$. Si $p \in A \cap X$, l'affirmation est bien vraie. Supposons donc que $p \in B \cap X$ (voir figure 8.5).

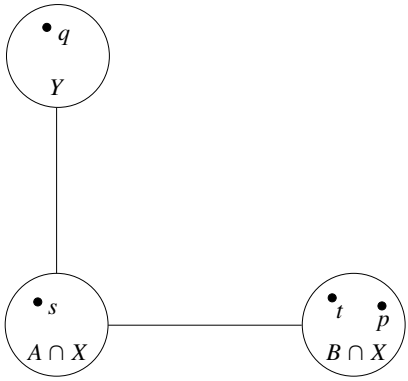


Figure 8.5.

Montrons que $\lambda_{sq} = \lambda_{pq}$. Puisque $\lambda_{pq} = w(e) = w(e')$ et $s \in A \cap X$, cela terminera la démonstration.

Par le lemme 8.30,

$$\lambda_{sq} \geq \min\{\lambda_{st}, \lambda_{tp}, \lambda_{pq}\}.$$

Puisque – par le lemme 8.33(b) – $E(A, B)$ est une coupe minimum séparant s et t, et puisque $s, q \in A$, nous pouvons déduire du lemme 8.32 que λ_{sq} ne change pas quand nous contractons B. Puisque $t, p \in B$, cela signifie que l'addition d'une arête (t, p) avec une capacité arbitrairement grande ne modifie pas λ_{sq}. Donc

$$\lambda_{sq} \geq \min\{\lambda_{st}, \lambda_{pq}\}.$$

Observons maintenant que $\lambda_{st} \geq \lambda_{pq}$ puisque la coupe minimum séparant s et t, $E(A, B)$, sépare également p et q. Donc

$$\lambda_{sq} \geq \lambda_{pq}.$$

Pour montrer l'égalité, observons que $w(e)$ est la capacité d'une coupe séparant X et Y, et donc s et q. Par conséquent

$$\lambda_{sq} \leq w(e) = \lambda_{pq}.$$

Cela termine la démonstration. □

Théorème 8.35. (Gomory et Hu [1961]) *L'*ALGORITHME DE GOMORY-HU *répond correctement. On peut associer à tout graphe non orienté un arbre de Gomory-Hu, et un tel arbre se construit en un temps* $O(n^3\sqrt{m})$.

Preuve. La complexité est égale à $n-1$ fois le temps nécessaire pour trouver une coupe minimum séparant s et t, puisque tout le reste peut s'implémenter en $O(n^3)$. Par la proposition 8.28 nous obtenons bien pour borne $O(n^3\sqrt{m})$.

Montrons que l'output T de l'algorithme est un arbre de Gomory-Hu pour (G,u). Il est clair que T est un arbre avec $V(T) = V(G)$. Soient $s,t \in V(G)$. Soit P_{st} l'unique chaîne de s à t de T et soient C_e et $V(G) \setminus C_e$ les composantes connexes de $T-e$ pour $e \in E(T)$.

Puisque $\delta(C_e)$ est une coupe séparant s et t pour chaque $e \in E(P_{st})$,

$$\lambda_{st} \leq \min_{e \in E(P_{st})} u(\delta(C_e)).$$

D'autre part, par une application répétée du lemme 8.30 nous avons :

$$\lambda_{st} \geq \min_{\{v,w\} \in E(P_{st})} \lambda_{vw}.$$

Donc en appliquant le lemme 8.34 à la situation avant l'exécution de ⑥ (où chaque sommet X de T est un singleton), nous avons

$$\lambda_{st} \geq \min_{e \in E(P_{st})} u(\delta(C_e)),$$

nous avons donc l'égalité. □

Un algorithme similaire pour ce problème (qui pourrait être plus facile à implémenter) a été suggéré par Gusfield [1990].

8.7 Capacité d'une coupe dans un graphe non orienté

Si nous recherchons la coupe de capacité minimum d'un graphe G non orienté ayant des capacités $u : E(G) \to \mathbb{R}_+$, il existe une méthode simple qui nécessite $n-1$ calculs de flot : fixer un sommet s et calculer la coupe minimum séparant s et t pour chaque $t \in V(G) \setminus \{s\}$. Il existe cependant des algorithmes plus efficaces.

Hao et Orlin [1994] ont trouvé un algorithme en $O(nm \log \frac{n^2}{m})$ pour la coupe de capacité minimum. Ils utilisent une version modifiée de l'ALGORITHME PUSH-RELABEL.

Si nous souhaitons calculer uniquement l'arête-connexité d'un graphe (quand les capacités valent 1), l'algorithme le plus rapide est celui de Gabow [1995] avec une complexité $O(m + \lambda^2 n \log \frac{n}{\lambda(G)})$, où $\lambda(G)$ est l'arête-connexité (observons que $2m \geq \lambda n$). L'algorithme de Gabow utilise des techniques d'intersection de matroïdes. Remarquons que le PROBLÈME DU FLOT MAXIMUM dans un graphe non orienté avec des capacités 1 peut se résoudre plus rapidement que dans le cas général (Karger et Levine [1998]).

Nagamochi et Ibaraki [1992] ont proposé un algorithme tout à fait différent pour trouver la coupe de capacité minimum dans un graphe non orienté. Leur algorithme ne fait pas du tout appel au calcul du flot maximum. Dans ce paragraphe, nous présentons leur méthode d'une manière simplifiée due à Stoer et Wagner [1997] et indépendamment à Frank [1994]. Nous commençons par une définition simple.

Définition 8.36. *Soit un graphe G ayant des capacités $u : E(G) \to \mathbb{R}_+$; un ordre $v_1, \dots, v_n$ des sommets sera appelé* **ordre MA (maximum adjacence)** *si pour tout $i \in \{2, \dots, n\}$:*

$$\sum_{e \in E(\{v_1,\dots,v_{i-1}\},\{v_i\})} u(e) \;=\; \max_{j \in \{i,\dots,n\}} \sum_{e \in E(\{v_1,\dots,v_{i-1}\},\{v_j\})} u(e).$$

Proposition 8.37. *Soit un graphe G ayant des capacités $u : E(G) \to \mathbb{R}_+$; un ordre MA peut être trouvé en un temps $O(m + n \log n)$.*

Preuve. Décrivons l'algorithme suivant. Posons d'abord $\alpha(v) := 0$ pour tout $v \in V(G)$. Puis pour $i := 1$ jusqu'à $i := n$ faisons ce qui suit : choisir v_i dans $V(G) \setminus \{v_1, \dots, v_{i-1}\}$ avec la plus grande valeur α (en cas d'égalité faire un choix arbitraire), et poser $\alpha(v) := \alpha(v) + \sum_{e \in E(\{v_i\},\{v\})} u(e)$ pour tout $v \in V(G) \setminus \{v_1, \dots, v_i\}$.

Cet algorithme répond correctement. En utilisant un tas de Fibonacci, attribuant à chaque sommet non encore choisi v une clé $-\alpha(v)$, nous obtenons un temps de calcul en $O(m+n \log n)$, par le théorème 6.6, puisqu'il y a n opérations d'INSERTION, n de SUPPRESSION et au plus m de DÉCROISSANCE. □

Lemme 8.38. (Stoer et Wagner [1997], Frank [1994]) *Soit G un graphe avec $n := |V(G)| \geq 2$, des capacités $u : E(G) \to \mathbb{R}_+$ et un ordre MA $v_1, \dots, v_n$. Alors*

$$\lambda_{v_{n-1}v_n} \;=\; \sum_{e \in E(\{v_n\},\{v_1,\dots,v_{n-1}\})} u(e).$$

Preuve. Il nous suffit de démontrer l'inégalité $\geq$, par induction sur $|V(G)| + |E(G)|$. Pour $|V(G)| < 3$ la relation $\geq$ est évidente. Nous pouvons supposer que $e = (v_{n-1}, v_n) \notin E(G)$, sinon on pourrait supprimer e (les valeurs des deux côtés de l'égalité de l'énoncé diminuent de $u(e)$) et appliquer l'hypothèse d'induction.

Appelons R le côté droit de l'égalité de l'énoncé. $v_1, \dots, v_{n-1}$ est un ordre MA dans $G - v_n$. Donc par induction,

$$\lambda^{G-v_n}_{v_{n-2}v_{n-1}} \;=\; \sum_{e \in E(\{v_{n-1}\},\{v_1,\dots,v_{n-2}\})} u(e) \;\geq\; \sum_{e \in E(\{v_n\},\{v_1,\dots,v_{n-2}\})} u(e) \;=\; R.$$

Ici l'inégalité est vérifiée parce que $v_1, \dots, v_n$ était un ordre MA sur G. La dernière égalité est vraie parce que $(v_{n-1}, v_n) \notin E(G)$. Donc $\lambda^G_{v_{n-2}v_{n-1}} \geq \lambda^{G-v_n}_{v_{n-2}v_{n-1}} \geq R$.

D'autre part $v_1, \dots, v_{n-2}, v_n$ est un ordre MA pour $G - v_{n-1}$. Par induction,

$$\lambda^{G-v_{n-1}}_{v_{n-2}v_n} \;=\; \sum_{e \in E(\{v_n\},\{v_1,\dots,v_{n-2}\})} u(e) \;=\; R,$$

parce que $(v_{n-1}, v_n) \notin E(G)$. Donc $\lambda^G_{v_{n-2}v_n} \geq \lambda^{G-v_{n-1}}_{v_{n-2}v_n} = R$.

Par le lemme 8.30 $\lambda_{v_{n-1}v_n} \geq \min\{\lambda_{v_{n-1}v_{n-2}}, \lambda_{v_{n-2}v_n}\} \geq R$. □

Notons que l'existence de deux sommets x, y avec $\lambda_{xy} = \sum_{e \in \delta(x)} u(e)$ avait été démontrée par Mader [1972], et se déduit facilement de l'existence d'un arbre de Gomory-Hu (exercice 25).

Théorème 8.39. (Nagamochi et Ibaraki [1992], Stoer et Wagner [1997]) *Une coupe de capacité minimum dans un graphe non orienté avec des capacités non négatives peut être trouvée en un temps* $O(mn + n^2 \log n)$.

Preuve. Nous pouvons supposer que G est simple puisqu'on peut agréger en une seule arête les arêtes parallèles entre elles. Notons par $\lambda(G)$ la capacité minimum d'une coupe de G. L'algorithme procède ainsi :

Soit $G_0 := G$. À l'étape i ($i = 1, \ldots, n-1$) choisir deux sommets $x, y \in V(G_{i-1})$ tels que

$$\lambda_{xy}^{G_{i-1}} = \sum_{e \in \delta_{G_{i-1}}(x)} u(e).$$

Par la proposition 8.37 et le lemme 8.38 cela peut être effectué en un temps $O(m + n \log n)$. Posons $\gamma_i := \lambda_{xy}^{G_{i-1}}$, $z_i := x$, et soit G_i le graphe résultant de G_{i-1} en contractant $\{x, y\}$. Notons que

$$\lambda(G_{i-1}) = \min\{\lambda(G_i), \gamma_i\}, \tag{8.1}$$

puisqu'une coupe minimum de G_{i-1} sépare x et y (sa capacité est alors γ_i) ou ne sépare pas ces deux sommets, et alors la contraction $\{x, y\}$ ne change rien.

Quand nous atteignons G_{n-1} qui n'a qu'un seul sommet, choisissons $k \in \{1, \ldots, n-1\}$ avec γ_k minimum. Si X est l'ensemble des sommets de G dont la contraction a créé le sommet z_k de G_{k-1}, $\delta(X)$ est une coupe de capacité minimum dans G. Ce dernier point est facile à voir puisque par (8.1), $\lambda(G) = \min\{\gamma_1, \ldots, \gamma_{n-1}\} = \gamma_k$ et γ_k est la capacité de la coupe $\delta(X)$. □

Un algorithme randomisé de contraction pour trouver une coupe minimum (avec une forte probabilité) est étudié dans l'exercice 29. Mentionnons également que la sommet-connexité d'un graphe peut se calculer par $O(n^2)$ calculs de flot (exercice 30).

Dans ce paragraphe, nous avons vu comment minimiser $f(X) := u(\delta(X))$ sur $\emptyset \neq X \subset V(G)$. Remarquons que $f : 2^{V(G)} \to \mathbb{R}_+$ est sous-modulaire et symétrique (c.-à-d. $f(A) = f(V(G) \setminus A)$ pour tout A). L'algorithme présenté ici a été généralisé par Queyranne [1998] pour minimiser des fonctions sous-modulaires symétriques générales ; voir le paragraphe 14.5.

Le problème de la recherche d'une coupe maximum est bien plus difficile et sera étudié au chapitre 16.2.

Exercices

1. Soit (G, u, s, t) un réseau, et soient $\delta^+(X)$ et $\delta^+(Y)$ deux coupes minimum séparant t de s dans (G, u). Montrer que $\delta^+(X \cap Y)$ et $\delta^+(X \cup Y)$ sont aussi des coupes minimum séparant t de s dans (G, u).

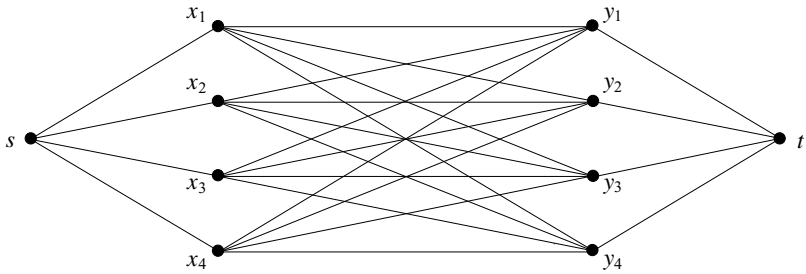


Figure 8.6.

2. Montrer que quand les capacités sont irrationnelles, l'ALGORITHME DE FORD-FULKERSON peut ne jamais terminer.
   Indication : étudier le réseau de la figure 8.6.
   Les segments de droite sur le dessin représentent des arcs dans les deux directions. Tous les arcs ont une capacité $S = \frac{1}{1-\sigma}$, à l'exception de

$$u((x_1, y_1)) = 1, \;\; u((x_2, y_2)) = \sigma, \;\; u((x_3, y_3)) = u((x_4, y_4)) = \sigma^2$$

   où $\sigma = \frac{\sqrt{5}-1}{2}$. Observons que $\sigma^n = \sigma^{n+1} + \sigma^{n+2}$.
   (Ford et Fulkerson [1962])

* 3. Soit G un graphe orienté et M sa matrice d'incidence. Montrer que pour tous $c, l, u \in \mathbb{Z}^{E(G)}$ avec $l \leq u$:

$$\max\left\{cx : x \in \mathbb{Z}^{E(G)},\, l \leq x \leq u,\, Mx = 0\right\} \;=$$

$$\min\left\{y'u - y''l :\, zM + y' + y" = c,\, y', y'' \in \mathbb{Z}_+^{E(G)},\, z \in \mathbb{Z}^{V(G)}\right\}.$$

   Montrer comment cela implique le théorème 8.6 et le corollaire 8.7.

4. Montrer le théorème de circulation de Hoffman : soit G un graphe orienté ayant des capacités supérieures et inférieures $l, u : E(G) \to \mathbb{R}_+$ vérifiant $l(e) \leq u(e)$ pour tout $e \in E(G)$: il existe une circulation f telle que $l(e) \leq f(e) \leq u(e)$ pour tout $e \in E(G)$ si et seulement si

$$\sum_{e \in \delta^-(X)} l(e) \;\leq\; \sum_{e \in \delta^+(X)} u(e) \quad \text{pour tout } X \subseteq V(G).$$

   Note : il est facile de montrer que le théorème de circulation de Hoffman implique le théorème max-flot/min-cut.
   (Hoffman [1960])

5. Soient un réseau (G, u, s, t), un flot maximum f de s à t et le graphe résiduel G_f. Construire le graphe orienté H obtenu à partir de G_f en contractant l'ensemble S des sommets connectés à s en un sommet v_S, puis en contractant l'ensemble T des sommets auxquels t est connecté en un sommet v_T et en contractant finalement chaque composante fortement connexe X de $G_f - (S \cup T)$ en

un sommet v_X ; H est sans circuit. Montrer qu'il existe une bijection entre les ensembles $X \subseteq V(G)$ pour lesquels $\delta^+_G(X)$ est une coupe minimum séparant t de s dans (G, u) et les ensembles $Y \subseteq V(H)$ pour lesquels $\delta^+_H(Y)$ est une coupe orientée séparant v_S de v_T dans H.
Note : cette condition reste valable dans G_f sans effectuer de contraction. Cependant, nous utiliserons la condition telle qu'elle est énoncée ici au paragraphe 20.4.
(Picard et Queyranne [1980])

6. Soit G un graphe orienté et soit $c : E(G) \to \mathbb{R}$. Nous cherchons un ensemble $X \subset V(G)$ avec $s \in X$ et $t \notin X$ tel que $\sum_{e\in\delta^+(X)} c(e) - \sum_{e\in\delta^-(X)} c(e)$ soit minimum. Montrer comment ramener ce problème au PROBLÈME DE LA COUPE DE CAPACITÉ MINIMUM.
Indication : construire un réseau où tous les arcs sont incidents à s ou t.

* 7. Soient G un graphe sans circuit, trois fonctions $\sigma, \tau, c : E(G) \to \mathbb{R}_+$, et un nombre $C \in \mathbb{R}_+$. Soit une fonction $x : E(G) \to \mathbb{R}_+$ telle que $\sigma(e) \leq x(e) \leq \tau(e)$ pour tout $e \in E(G)$ et $\sum_{e\in E(G)}(\tau(e) - x(e))c(e) \leq C$. Nous cherchons à déterminer x de telle sorte que la longueur du plus long chemin (x étant la fonction coût sur les arcs) de G soit minimum.

La signification est la suivante : les arcs correspondent aux tâches, $\sigma(e)$ et $\tau(e)$ représentent respectivement le temps minimum et maximum d'exécution de la tâche e et $c(e)$ est le coût unitaire de réduction de la tâche e. Si $e = (i, j)$ et $e' = (j, k)$, cela signifie que la tâche e doit être terminée avant de pouvoir démarrer la tâche e'. Nous avons un budget fixé C et nous voulons minimiser le temps d'exécution de toutes les tâches.

Montrer comment on peut résoudre ce problème par des techniques de flots dans les réseaux. (Cette application est connue sous le nom de méthode PERT, «*program evaluation and review technique*» ou MCC, méthode du chemin critique.)

Indication : introduire une source s et un puits t. Commencer avec $x = \tau$ et réduire successivement la longueur du plus long chemin de s à t (calculé avec les coûts x), jusqu'à atteindre la valeur minimum. Utiliser l'exercice 7 du chapitre 7, l'exercice 8 du chapitre 3, et l'exercice 6.
(Phillips et Dessouky [1977])

* 8. Soit (G, c, s, t) un réseau tel que G soit planaire, même avec l'adjonction de l'arc $e = (s, t)$. Décrivons l'algorithme suivant. Partons du flot $f \equiv 0$ et posons $G' := G_f$. À chaque étape considérons la frontière B d'une face de $G' + e$ contenant e (dans une représentation plane fixée). Augmenter f le long de $B-e$. Soit G' le graphe constitué seulement des arcs directs de G_f et itérer aussi longtemps que t est connecté à s dans G'.

Montrer que cet algorithme calcule un flot de s à t maximum. Utiliser le théorème 2.40 pour montrer que la complexité de cet algorithme est $O(n^2)$.
(Ford et Fulkerson [1956], Hu [1969])
Note : ce problème peut être résolu en un temps $O(n)$. Pour les réseaux pla-

naires en général il existe un algorithme en $O(n \log n)$; voir Weihe [1997] et Borradaile et Klein [2006].

9. Montrer que la version arc-disjoint du théorème de Menger 8.9 se déduit directement du théorème 6.17.

10. Soient G un graphe orienté (resp. non orienté), x, y, z trois sommets, et $\alpha, \beta \in \mathbb{N}$ avec $\alpha \leq \lambda_{xy}$, $\beta \leq \lambda_{xz}$ et $\alpha + \beta \leq \max\{\lambda_{xy}, \lambda_{xz}\}$. Montrer qu'il existe α chemins (resp. chaînes) de x à y et β chemins (resp. chaînes) de x à z de telle sorte que ces $\alpha + \beta$ chemins (resp. chaînes) soient deux à deux arc-disjoints (resp. arête-disjointes).

11. Soit G un graphe orienté contenant k chemins arc-disjoints de s à t pour toute paire de sommets s et t (un tel graphe est appelé k-arc-connexe).
Soit H un graphe orienté avec $V(H) = V(G)$ et $|E(H)| = k$. Montrer que l'instance (G, H) du PROBLÈME DES CHEMINS ARC-DISJOINTS a une solution.
(Mader [1981] et Shiloach [1979])

12. Soit G un graphe orienté ayant au moins k arcs. Montrer que : G contient k chemins de s à t arc-disjoints pour toute paire de sommets s et t si et seulement si, pour tout ensemble de k arcs distincts $e_1 = (x_1, y_1), \ldots, e_k = (x_k, y_k)$, $G - \{e_1, \ldots, e_k\}$ contient k arborescences couvrantes arc-disjointes $T_1, \ldots, T_k$ telles que T_i soit enracinée en y_i $(i = 1, \ldots, k)$.
Note : cela généralise l'exercice 11.
Indication : utiliser le théorème 6.17.
(Su [1997])

13. Soit G un graphe orienté avec des capacités $c : E(G) \to \mathbb{R}_+$ et $r \in V(G)$. Peut-on trouver une coupe issue de r de capacité minimum en temps polynomial ? Peut-on trouver une coupe orientée de capacité minimum en temps polynomial (ou décider que G est fortement connexe) ?
Note : la réponse à la première question résout le PROBLÈME DE SÉPARATION pour le PROBLÈME DE L'ARBORESCENCE ENRACINÉE DE POIDS MINIMUM ; voir le corollaire 6.14.

14. Montrer comment trouver un flot dans un réseau sans circuit en un temps $O(nm)$.
(Dinic [1970])

15. Soit (G, u, s, t) un réseau tel que $G - t$ soit une arborescence. Montrer comment trouver un flot maximum de s à t en temps linéaire.
Indication : utiliser la procédure DFS.

* 16. Soit (G, u, s, t) un réseau tel que le graphe non orienté de $G - \{s, t\}$ soit une forêt. Montrer comment trouver un flot maximum de s à t en temps linéaire.
(Vygen [2002])

17. Voici une variante de l'ALGORITHME DE FUJISHIGE où dans ⑤ nous choisissons $v_i \in V(G) \setminus \{v_1, \ldots, v_{i-1}\}$ tel que $b(v_i)$ soit maximum, où l'étape ④ est remplacée par stop si $b(v) = 0$ pour tout $v \in V(G) \setminus \{v_1, \ldots, v_i\}$, et où

dans le début de ⑥ nous posons $\beta(t) := \min_{j=2}^{i} b(j)$. X et α ne sont alors plus nécessaires.

(a) Montrer que cette variante répond correctement.

(b) Soit α_k le nombre $\min_{j=2}^{i} b(j)$ à l'itération k (ou zéro si l'algorithme stoppe avant l'itération k). Montrer que $\min_{l=k+1}^{k+2n} \alpha_l \leq \frac{1}{2}\alpha_k$ pour tout k. Conclure que le nombre d'itérations est $O(n \log u_{\max})$.

(c) Montrer comment implémenter une itération en un temps $O(m + n \log n)$.

18. Nous dirons qu'un pré-flot f est maximum si $\mathrm{ex}_f(t)$ est maximum.

(a) Montrer que, si f est un pré-flot maximum, il existe un flot maximum f' avec $f'(e) \leq f(e)$ pour tout $e \in E(G)$.

(b) Montrer qu'un pré-flot maximum peut être transformé en un flot maximum en un temps $O(nm)$. (*Indication :* utiliser une variante de l'ALGORITHME D'EDMONDS-KARP.)

19. Montrer que l'ALGORITHME PUSH-RELABEL crée $O(n^2 m)$ push non saturés, indépendamment du choix de v dans ③.

20. Soit un graphe sans circuit G avec des poids $c : E(G) \to \mathbb{R}_+$; trouver une coupe orientée de poids maximum dans G. Montrer comment ce problème se ramène au calcul d'une coupe minimum séparant t de s et peut se résoudre en un temps $O(n^3)$.
Indication : utiliser l'exercice 6.

21. Soit G un graphe orienté sans circuit avec des poids $c : E(G) \to \mathbb{R}_+$. Nous cherchons le poids maximum d'un sous-ensemble $F \subseteq E(G)$ tel qu'aucun chemin de G ne contienne plus qu'un arc de F. Montrer que ce problème est équivalent à la recherche d'une coupe orientée de poids maximum dans G (et peut être résolu en un temps $O(n^3)$ par l'exercice 20).

22. Soit G un graphe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$ et un ensemble $T \subseteq V(G)$ avec $|T| \geq 2$. Nous cherchons un ensemble $X \subset V(G)$ avec $T \cap X \neq \emptyset$ et $T \setminus X \neq \emptyset$ tel que $\sum_{e \in \delta(X)} u(e)$ soit minimum. Montrer comment résoudre ce problème en un temps $O(n^4)$ ($n = |V(G)|$).

23. Soient λ_{ij}, $1 \leq i, j \leq n$, des nombres non négatifs avec $\lambda_{ij} = \lambda_{ji}$ et $\lambda_{ik} \geq \min(\lambda_{ij}, \lambda_{jk})$ pour tout ensemble de trois indices distincts $i, j, k \in \{1, \ldots, n\}$. Montrer qu'il existe un graphe G avec $V(G) = \{1, \ldots, n\}$ et des capacités $u : E(G) \to \mathbb{R}_+$ tels que les arc-connexités soient précisément les λ_{ij}.
Indication : considérer un arbre couvrant maximum (K_n, c), où $c((i,j)) := \lambda_{ij}$.
(Gomory et Hu [1961])

24. Soit G un graphe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$, et soit $T \subseteq V(G)$ avec $|T|$ pair. Une T-coupe dans G est une coupe $\delta(X)$ avec $|X \cap T|$ impair. Proposer un algorithme polynomial pour trouver la T-coupe de capacité minimum dans (G, u).
Indication : utiliser l'arbre de Gomory-Hu.
(Une solution de cet exercice sera donnée au paragraphe 12.3.)

25. Soit G un graphe simple non orienté avec au moins deux sommets. Supposons que le degré de chaque sommet de G soit au moins k. Montrer qu'il existe deux sommets s et t connectés par k chaînes de s à t arête-disjointes. Que se passe-t-il s'il existe un seul sommet de degré plus petit que k ?
Indication : considérer l'arbre de Gomory-Hu dans G.

26. Nous cherchons à connaître l'arête-connexité $\lambda(G)$ d'un graphe non orienté (les capacités sont égales à 1). Le paragraphe 8.7 montre comment résoudre ce problème en un temps $O(mn)$, pourvu qu'on puisse trouver un ordre MA d'un graphe non orienté avec des capacités 1 en un temps $O(m+n)$. Comment résoudre ce problème ?

∗ 27. Soit G un graphe non orienté avec un ordre MA $v_1, \ldots, v_n$. Soit κ_{uv} le nombre maximum de chaînes arête-disjointes de u à v. Montrer que $\kappa_{v_{n-1}v_n} = |E(\{v_n\}, \{v_1, \ldots, v_{n-1}\})|$ (la contrepartie sommet-disjoint du lemme 8.38).
Indication : montrer par induction que $\kappa^{G_{ij}}_{v_j v_i} = |E(\{v_j\}, \{v_1, \ldots, v_i\})|$, où $G_{ij} = G[\{v_1, \ldots, v_i\} \cup \{v_j\}]$. Pour cela , on peut toujours supposer que $(v_j, v_i) \notin E(G)$, choisir un ensemble minimal $Z \subseteq \{v_1, \ldots, v_{i-1}\}$ séparant v_j et v_i (théorème de Menger 8.10), et soit $h \leq i$ le nombre maximum tel que $v_h \notin Z$ et tel que v_h soit adjacent à v_i ou v_j.
(Frank [non publié])

∗ 28. Un graphe non orienté est appelé triangulé s'il n'a pas de cycle de taille de longueur supérieure ou égale à quatre comme sous-graphe induit. Un ordre $v_1, \ldots, v_n$ d'un graphe non orienté G est dit simplicial si $(v_i, v_j), (v_i, v_k) \in E(G)$ implique $(v_j, v_k) \in E(G)$ pour $i < j < k$.

(a) Montrer qu'un graphe avec un ordre simplicial est triangulé.

(b) Soit G un graphe triangulé, et soit $v_1, \ldots, v_n$ un ordre MA. Montrer que $v_n, v_{n-1}, \ldots, v_1$ est un ordre simplicial.
Indication : utiliser l'exercice 27 et le théorème de Menger 8.10.

Note : la propriété qu'un graphe est triangulé si et seulement s'il admet un ordre simplicial est due à Rose [1970].

29. Soit G un graphe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$. Soit $\emptyset \neq A \subset V(G)$ tel que $\delta(A)$ soit une coupe de capacité minimum dans G.

(a) Montrer que $u(\delta(A)) \leq \frac{2}{n} u(E(G))$. (*Indication :* étudier les coupes triviales $\delta(x)$, $x \in V(G)$.)

(b) Considérons la procédure suivante : choisissons une arête aléatoirement que nous contractons, chaque arête e étant choisie avec une probabilité $\frac{u(e)}{u(E(G))}$. Répétons cette opération jusqu'à ce qu'il ne reste plus que deux sommets. Montrer que la probabilité de ne jamais contracter une arête de $\delta(A)$ est au moins $\frac{2}{(n-1)n}$.

(c) En conclure que, si on exécute l'algorithme randomisé de (b) kn^2 fois, on trouve $\delta(A)$ avec une probabilité supérieure ou égale à $1 - e^{-2k}$. (Un tel algorithme qui fournit une bonne réponse avec une probabilité positive est appelé algorithme de Monte-Carlo.)

(Karger et Stein [1996] ; voir aussi Karger [2000])

30. Montrer comment la sommet-connexité d'un graphe non orienté peut se calculer en un temps $O(n^5)$.
Indication : on reverra la preuve du théorème de Menger.
Note : il existe un algorithme en $O(n^4)$; voir Henzinger, Rao et Gabow [2000].

31. Soit G un graphe connexe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$. Nous cherchons une 3-coupe de capacité minimum, c.-à-d. un ensemble d'arêtes dont la suppression sépare G en au moins trois composantes connexes. Soit $n := |V(G)| \geq 4$. Soient $\delta(X_1), \delta(X_2), \ldots$ une liste de coupes ordonnées par capacités non décroissantes : $u(\delta(X_1)) \leq u(\delta(X_2)) \leq \cdots$. Supposons que nous connaissions les $2n-2$ éléments de cette liste (*note* : ces éléments peuvent être calculés en temps polynomial par une méthode de Vazirani et Yannakakis [1992]).
    (a) Montrer qu'il existe des indices $i, j \in \{1, \ldots, 2n-2\}$ tels que les ensembles $X_i \setminus X_j$, $X_j \setminus X_i$, $X_i \cap X_j$ et $V(G) \setminus (X_i \cup X_j)$ soient non vides.
    (b) Montrer qu'il existe une 3-coupe de capacité au plus $\frac{3}{2}u(\delta(X_{2n-2})$.
    (c) Pour chaque $i = 1, \ldots, 2n-2$ soit l'union de $\delta(X_i)$ et d'une coupe minimum de $G - X_i$, et aussi l'union de $\delta(X_i)$ et d'une coupe minimum de $G[X_i]$. Cela donne une liste de, au plus, $4n-4$ 3-coupes. Montrer que l'une d'entre elles est minimum.

    (Nagamochi et Ibaraki [2000])
    Note : le problème de la 3-coupe minimum séparant trois sommets donnés est bien plus difficile ; voir Dahlhaus *et al.* [1994] et Cheung, Cunningham et Tang [2006].

Références

Littérature générale :

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. [1993] : Network Flows. Prentice-Hall, Englewood Cliffs 1993

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. [1998] : Combinatorial Optimization. Wiley, New York 1998, Chapter 3

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. [2001] : Introduction to Algorithms. Second Edition. MIT Press, Cambridge 2001, Chapter 26

Ford, L.R., Fulkerson, D.R. [1962] : Flows in Networks. Princeton University Press, Princeton 1962

Frank, A. [1995] : Connectivity and network flows. In : Handbook of Combinatorics ; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam, 1995

Goldberg, A.V., Tardos, É., Tarjan, R.E. [1990] : Network flow algorithms. In : Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, eds.), Springer, Berlin 1990, pp. 101–164

Gondran, M., Minoux, M. [1984] : Graphs and Algorithms. Wiley, Chichester 1984, Chapter 5

Jungnickel, D. [1999] : Graphs, Networks and Algorithms. Springer, Berlin 1999

Phillips, D.T., Garcia-Diaz, A. [1981] : Fundamentals of Network Analysis. Prentice-Hall, Englewood Cliffs 1981

Ruhe, G. [1991] : Algorithmic Aspects of Flows in Networks. Kluwer Academic Publishers, Dordrecht 1991

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 9,10,13–15

Tarjan, R.E. [1983] : Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Chapter 8

Thulasiraman, K., Swamy, M.N.S. [1992] : Graphs : Theory and Algorithms. Wiley, New York 1992, Chapter 12

Références citées :

Ahuja, R.K., Orlin, J.B., Tarjan, R.E. [1989] : Improved time bounds for the maximum flow problem. SIAM Journal on Computing 18 (1989), 939–954

Borradaile, G. Klein, P. [2006] : An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (2006), 524–533

Cheriyan, J., Maheshwari, S.N. [1989] : Analysis of preflow push algorithms for maximum network flow. SIAM Journal on Computing 18 (1989), 1057–1086

Cheriyan, J., Mehlhorn, K. [1999] : An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. Information Processing Letters 69 (1999), 239–242

Cherkassky, B.V. [1977] : Algorithm of construction of maximal flow in networks with complexity of $O(V^2\sqrt{E})$ operations. Mathematical Methods of Solution of Economical Problems 7 (1977), 112–125 [in Russian]

Cheung, K.K.H., Cunningham, W.H., Tang, L. [2006] : Optimal 3-terminal cuts and linear programming. Mathematical Programming 106 (2006), 1–23

Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M. [1994] : The complexity of multiterminal cuts. SIAM Journal on Computing 23 (1994), 864–894

Dantzig, G.B., Fulkerson, D.R. [1956] : On the max-flow min-cut theorem of networks. In : Linear Inequalities and Related Systems (H.W. Kuhn, A.W. Tucker, eds.), Princeton University Press, Princeton 1956, pp. 215–221

Dinic, E.A. [1970] : Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Mathematics Doklady 11 (1970), 1277–1280

Edmonds, J., Karp, R.M. [1972] : Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM 19 (1972), 248–264

Elias, P., Feinstein, A., Shannon, C.E. [1956] : Note on maximum flow through a network. IRE Transactions on Information Theory, IT-2 (1956), 117–119

Ford, L.R., Fulkerson, D.R. [1956] : Maximal Flow Through a Network. Canadian Journal of Mathematics 8 (1956), 399–404

Ford, L.R., Fulkerson, D.R. [1957] : A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. Canadian Journal of Mathematics 9 (1957), 210–218

Frank, A. [1994] : On the edge-connectivity algorithm of Nagamochi and Ibaraki. Laboratoire Artemis, IMAG, Université J. Fourier, Grenoble, 1994

Fujishige, S. [2003] : A maximum flow algorithm using MA ordering. Operations Research Letters 31 (2003), 176–178

Gabow, H.N. [1995] : A matroid approach to finding edge connectivity and packing arborescences. Journal of Computer and System Sciences 50 (1995), 259–273

Galil, Z. [1980] : An $O(V^{\frac{5}{3}}E^{\frac{2}{3}})$ algorithm for the maximal flow problem. Acta Informatica 14 (1980), 221–242

Galil, Z., Namaad, A. [1980] : An $O(EV\log^2 V)$ algorithm for the maximal flow problem. Journal of Computer and System Sciences 21 (1980), 203–217

Gallai, T. [1958] : Maximum-minimum Sätze über Graphen. Acta Mathematica Academiae Scientiarum Hungaricae 9 (1958), 395–434

Goldberg, A.V., Rao, S. [1998] : Beyond the flow decomposition barrier. Journal of the ACM 45 (1998), 783–797

Goldberg, A.V., Tarjan, R.E. [1988] : A new approach to the maximum flow problem. Journal of the ACM 35 (1988), 921–940

Gomory, R.E., Hu, T.C. [1961] : Multi-terminal network flows. Journal of SIAM 9 (1961), 551–570

Gusfield, D. [1990] : Very simple methods for all pairs network flow analysis. SIAM Journal on Computing 19 (1990), 143–155

Hao, J., Orlin, J.B. [1994] : A faster algorithm for finding the minimum cut in a directed graph. Journal of Algorithms 17 (1994), 409–423

Henzinger, M.R., Rao, S., Gabow, H.N. [2000] : Computing vertex connectivity : new bounds from old techniques. Journal of Algorithms 34 (2000), 222–250

Hoffman, A.J. [1960] : Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. In : Combinatorial Analysis (R.E. Bellman, M. Hall, eds.), AMS, Providence 1960, pp. 113–128

Hu, T.C. [1969] : Integer Programming and Network Flows. Addison-Wesley, Reading 1969

Karger, D.R. [2000] : Minimum cuts in near-linear time. Journal of the ACM 47 (2000), 46–76

Karger, D.R., Levine, M.S. [1998] : Finding maximum flows in undirected graphs seems easier than bipartite matching. Proceedings of the 30th Annual ACM Symposium on Theory of Computing (1998), 69–78

Karger, D.R., Stein, C. [1996] : A new approach to the minimum cut problem. Journal of the ACM 43 (1996), 601–640

Karzanov, A.V. [1974] : Determining the maximal flow in a network by the method of preflows. Soviet Mathematics Doklady 15 (1974), 434–437

King, V., Rao, S., Tarjan, R.E. [1994] : A faster deterministic maximum flow algorithm. Journal of Algorithms 17 (1994), 447–474

Mader, W. [1972] : Über minimal n-fach zusammenhängende, unendliche Graphen und ein Extremalproblem. Arch. Math. 23 (1972), 553–560

Mader, W. [1981] : On a property of n edge-connected digraphs. Combinatorica 1 (1981), 385–386

Malhotra, V.M., Kumar, M.P., Maheshwari, S.N. [1978] : An $O(|V|^3)$ algorithm for finding maximum flows in networks. Information Processing Letters 7 (1978), 277–278

Menger, K. [1927] : Zur allgemeinen Kurventheorie. Fundamenta Mathematicae 10 (1927), 96–115

Nagamochi, H., Ibaraki, T. [1992] : Computing edge-connectivity in multigraphs and capacitated graphs. SIAM Journal on Discrete Mathematics 5 (1992), 54–66

Nagamochi, H., Ibaraki, T. [2000] : A fast algorithm for computing minimum 3-way and 4-way cuts. Mathematical Programming 88 (2000), 507–520

Phillips, S., Dessouky, M.I. [1977] : Solving the project time/cost tradeoff problem using the minimal cut concept. Management Science 24 (1977), 393–400

Picard, J., Queyranne, M. [1980] : On the structure of all minimum cuts in a network and applications. Mathematical Programming Study 13 (1980), 8–16

Queyranne, M. [1998] : Minimizing symmetric submodular functions. Mathematical Programming B 82 (1998), 3–12

Rose, D.J. [1970] : Triangulated graphs and the elimination process. Journal of Mathematical Analysis and Applications 32 (1970), 597–609

Shiloach, Y. [1978] : An $O(nI \log^2 I)$ maximum-flow algorithm. Technical Report STAN-CS-78-802, Computer Science Department, Stanford University, 1978

Shiloach, Y. [1979] : Edge-disjoint branching in directed multigraphs. Information Processing Letters 8 (1979), 24–27

Shioura, A. [2004] : The MA ordering max-flow algorithm is not strongly polynomial for directed networks. Operations Research Letters 32 (2004), 31–35

Sleator, D.D. [1980] : An $O(nm \log n)$ algorithm for maximum network flow. Technical Report STAN-CS-80-831, Computer Science Department, Stanford University, 1978

Sleator, D.D., Tarjan, R.E. [1983] : A data structure for dynamic trees. Journal of Computer and System Sciences 26 (1983), 362–391

Su, X.Y. [1997] : Some generalizations of Menger's theorem concerning arc-connected digraphs. Discrete Mathematics 175 (1997), 293–296

Stoer, M., Wagner, F. [1997] : A simple min cut algorithm. Journal of the ACM 44 (1997), 585–591

Tunçel, L. [1994] : On the complexity preflow-push algorithms for maximum flow problems. Algorithmica 11 (1994), 353–359

Vazirani, V.V., Yannakakis, M. [1992] : Suboptimal cuts : their enumeration, weight, and number. In : Automata, Languages and Programming ; Proceedings ; LNCS 623 (W. Kuich, ed.), Springer, Berlin 1992, pp. 366–377

Vygen, J. [2002] : On dual minimum cost flow algorithms. Mathematical Methods of Operations Research 56 (2002), 101–126

Weihe, K. [1997] : Maximum (s, t)-flows in planar networks in $O(|V| \log |V|)$ time. Journal of Computer and System Sciences 55 (1997), 454–475

Whitney, H. [1932] : Congruent graphs and the connectivity of graphs. American Journal of Mathematics 54 (1932), 150–168

Chapitre 9

Flots de coût minimum

Nous étudierons dans ce chapitre des problèmes de flot quand des coûts sont affectés aux arcs du réseau. Par exemple, si on associe aux arcs du réseau du PROBLÈME D'AFFECTATION DES TÂCHES (qui peut se formuler comme un problème de flot ; voir l'introduction du chapitre 8) des coûts représentant le salaire des employés, notre objectif pourra consister à trouver une affectation de coût minimum permettant d'effectuer toutes les tâches dans un temps donné. Le problème que nous étudierons dans ce chapitre a de nombreuses autres applications.

Nous supposerons également que le réseau a plusieurs sources et puits. Nous présenterons le problème général et un cas particulier important au paragraphe 9.1. Au paragraphe 9.2, nous étudierons les critères d'optimalité qui forment la base des algorithmes de flots de coût minimum présentés aux paragraphes 9.3, 9.4, 9.5 et 9.6. La plupart de ces algorithmes se servent des méthodes étudiées au chapitre 7 pour trouver un circuit moyen minimum ou un plus court chemin. Le paragraphe 9.7 conclut ce chapitre par une application aux flots dynamiques.

9.1 Formulation du problème

Les données seront ici un graphe orienté G, des capacités $u : E(G) \to \mathbb{R}_+$, et des nombres $c : E(G) \to \mathbb{R}$ représentant le coût des arcs. De plus, nous autoriserons plusieurs sources et puits :

Définition 9.1. *Soient un graphe orienté G, des capacités $u : E(G) \to \mathbb{R}_+$, et des nombres $b : V(G) \to \mathbb{R}$ avec $\sum_{v \in V(G)} b(v) = 0$, Un **$b$-flot** dans (G, u) est une fonction $f : E(G) \to \mathbb{R}_+$ telle que $f(e) \leq u(e)$ pour tout $e \in E(G)$ et $\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v)$ pour tout $v \in V(G)$.*

Un b-flot avec $b \equiv 0$ est une circulation. $b(v)$ est la **valeur** du sommet v. $|b(v)|$ est quelquefois appelé l'**offre** (si $b(v) > 0$) ou la **demande** (si $b(v) < 0$) de v. Les sommets v avec $b(v) > 0$ sont appelés **sources**, ceux avec $b(v) < 0$ **puits**.

Remarquons que tout algorithme de résolution du PROBLÈME DU FLOT MAXIMUM permet de trouver un b-flot : ajoutons en effet deux sommets s, t ainsi que les arcs (s,v), (v,t) munis de capacités $u((s,v)) := \max\{0, b(v)\}$ et $u((v,t)) := \max\{0, -b(v)\}$ pour tout $v \in V(G)$. Alors, tout flot de s à t ayant pour valeur $\sum_{v \in V(G)} u((s,v))$ dans ce réseau fournit un b-flot dans G. Donc un critère pour l'existence d'un b-flot peut se déduire du théorème flot-max/coupe-min 8.6 (voir exercice 2). Le problème consiste à trouver un b-flot de coût minimum :

PROBLÈME DU FLOT DE COÛT MINIMUM

Instance Un graphe orienté G, des capacités $u : E(G) \to \mathbb{R}_+$, des nombres $b : V(G) \to \mathbb{R}$ avec $\sum_{v \in V(G)} b(v) = 0$, et des coûts $c : E(G) \to \mathbb{R}$.

Tâche Trouver un b-flot f dont le coût $c(f) := \sum_{e \in E(G)} f(e)c(e)$ est minimum (ou conclure qu'aucun b-flot n'existe).

Nous autoriserons parfois des capacités infinies. Dans ce cas une instance pourrait être non bornée, mais cela peut se vérifier à l'avance ; voir l'exercice 5.

Le PROBLÈME DU FLOT DE COÛT MINIMUM est tout à fait général et inclut des problèmes spécifiques intéressants. Le cas sans capacités ($u \equiv \infty$) est quelquefois appelé le problème du transbordement. Un problème encore plus particulier, connu sous le nom de problème de transport, a été formulé il y a longtemps par Hitchcock [1941] et d'autres auteurs :

PROBLÈME DE HITCHCOCK

Instance Un graphe orienté G avec $V(G) = A \dot\cup B$ et $E(G) \subseteq A \times B$. Des offres $b(v) \geq 0$ pour $v \in A$ et des demandes $-b(v) \geq 0$ pour $v \in B$ avec $\sum_{v \in V(G)} b(v) = 0$. Des coûts $c : E(G) \to \mathbb{R}$.

Tâche Trouver un b-flot f dans (G, ∞) de coût minimum (ou décider qu'aucun b-flot n'existe).

On peut toujours supposer dans le PROBLÈME DE HITCHCOCK que les coûts c sont non négatifs : en effet en ajoutant une constante α à tous les coûts on augmente le coût de chaque b-flot d'une même quantité, à savoir $\alpha \sum_{v \in A} b(v)$. On étudie donc souvent le cas où c est non négatif et $E(G) = A \times B$.

Il est évident que toute instance du PROBLÈME DE HITCHCOCK s'écrit comme une instance du PROBLÈME DU FLOT DE COÛT MINIMUM sur un graphe biparti avec des capacités infinies. Il est moins évident que toute instance du PROBLÈME DU FLOT DE COÛT MINIMUM peut être transformée en une instance équivalente (mais de taille plus grande) du PROBLÈME DE HITCHCOCK :

Lemme 9.2. (Orden [1956], Wagner [1959]) *Une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM *avec n sommets et m arcs peut être transformée en une instance équivalente du* PROBLÈME DE HITCHCOCK *avec $n + m$ sommets et $2m$ arcs.*

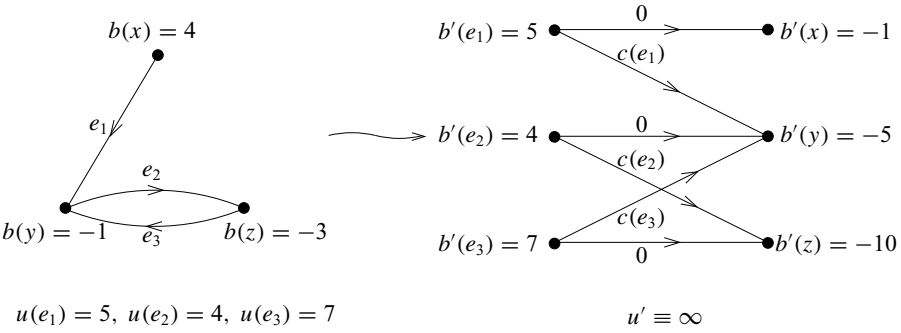


Figure 9.1.

Preuve. Soit (G, u, b, c) une instance du PROBLÈME DU FLOT DE COÛT MINIMUM. Définissons l'instance (G', A', B', b', c') du PROBLÈME DE HITCHCOCK :

Soit $A' := E(G)$, $B' := V(G)$ et $G' := (A' \cup B', E_1 \cup E_2)$, où $E_1 := \{((x,y),x) : (x,y) \in E(G)\}$ et $E_2 := \{((x,y),y) : (x,y) \in E(G)\}$. Soit $c'((e,x)) := 0$ pour $(e,x) \in E_1$ et $c'((e,y)) := c(e)$ pour $(e,y) \in E_2$. Enfin soit $b'(e) := u(e)$ pour $e \in E(G)$ et

$$b'(x) := b(x) - \sum_{e \in \delta_G^+(x)} u(e) \quad \text{pour } x \in V(G).$$

Un exemple est illustré par la figure 9.1.

Montrons que les deux instances sont équivalentes. Soit f un b-flot dans (G,u). Posons $f'((e,y)) := f(e)$ et $f'((e,x)) := u(e) - f(e)$ pour $e = (x,y) \in E(G)$. f' est bien un b'-flot de G' avec $c'(f') = c(f)$.

Inversement, si f' est un b'-flot de G', alors $f((x,y)) := f'(((x,y),y))$ est un b-flot de G avec $c(f) = c'(f')$. □

La preuve précédente est due à Ford et Fulkerson [1962].

9.2 Un critère d'optimalité

Nous allons montrer dans ce paragraphe quelques résultats simples, en particulier un critère d'optimalité qui sera la base des algorithmes des paragraphes suivants. Nous utiliserons de nouveau les concepts de graphes résiduels et de chemins augmentants. Nous étendrons les coûts c à $\overleftrightarrow{G}$ en posant $c(\overleftarrow{e}) := -c(e)$ pour tout arc $e \in E(G)$. L'avantage de cette définition est que le coût d'un arc dans le graphe résiduel G_f est indépendant du flot f.

Définition 9.3. *Si G est un graphe orienté muni de capacités et f est un b-flot, un* **circuit f-augmentant** *est un circuit de G_f.*

L'observation suivante sera utile :

Proposition 9.4. *Soit G un graphe orienté avec des capacités $u : E(G) \to \mathbb{R}_+$. Soient f et f' des b-flots dans (G, u). Alors $g : E(\overleftrightarrow{G}) \to \mathbb{R}_+$ défini par $g(e) := \max\{0, f'(e) - f(e)\}$ et $g(\overleftarrow{e}) := \max\{0, f(e) - f'(e)\}$ pour $e \in E(G)$ est une circulation dans $\overleftrightarrow{G}$. De plus, $g(e) = 0$ pour tout $e \notin E(G_f)$ et $c(g) = c(f') - c(f)$.*

Preuve. En chaque sommet $v \in V(\overleftrightarrow{G})$

$$\begin{aligned}\sum_{e \in \delta^+_{\overleftrightarrow{G}}(v)} g(e) - \sum_{e \in \delta^-_{\overleftrightarrow{G}}(v)} g(e) &= \sum_{e \in \delta^+_G(v)} (f'(e) - f(e)) - \sum_{e \in \delta^-_G(v)} (f'(e) - f(e)) \\ &= b(v) - b(v) = 0,\end{aligned}$$

donc g est une circulation dans $\overleftrightarrow{G}$.

Pour $e \in E(\overleftrightarrow{G}) \setminus E(G_f)$ nous étudierons deux cas : si $e \in E(G)$ alors $f(e) = u(e)$ et $f'(e) \leq f(e)$, ce qui implique $g(e) = 0$. Si $e = \overleftarrow{e_0}$ pour un arc $e_0 \in E(G)$ alors $f(e_0) = 0$ et donc $g(\overleftarrow{e_0}) = 0$.

La dernière condition de l'énoncé est facilement vérifiée :

$$c(g) = \sum_{e \in E(\overleftrightarrow{G})} c(e)g(e) = \sum_{e \in E(G)} c(e)f'(e) - \sum_{e \in E(G)} c(e)f(e) = c(f') - c(f).$$

□

Les circulations peuvent se décomposer en flots portés par des circuits de la même manière que les graphes eulériens se partitionnent en circuits :

Proposition 9.5. (Ford et Fulkerson [1962]) *Si f est une circulation d'un graphe orienté G, il existe une famille $\mathcal{C}$ avec au plus $|E(G)|$ circuits dans G et des nombres positifs $h(C)$ ($C \in \mathcal{C}$) tels que $f(e) = \sum\{h(C) : C \in \mathcal{C}, e \in E(C)\}$ pour tout $e \in E(G)$.*

Preuve. C'est un cas particulier du théorème 8.8. □

Donnons maintenant un critère d'optimalité :

Théorème 9.6. (Klein [1967]) *Soit (G, u, b, c) une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM. *Un b-flot f est de coût minimum si et seulement s'il n'existe pas de circuit f-augmentant de coût total négatif.*

Preuve. S'il existe un circuit f-augmentant C de coût total $\gamma < 0$, nous pouvons augmenter f le long de C par $\varepsilon > 0$ et obtenir un b-flot f' de coût diminué de $-\gamma\varepsilon$. Donc f n'est pas un flot de coût minimum.

Si f n'est pas un b-flot de coût minimum, il existe un b-flot f' de coût plus petit. Soit g défini comme dans la proposition 9.4 ; g est alors une circulation avec $c(g) < 0$. Par la proposition 9.5, g se décompose en flots ayant comme support des

circuits. Puisque $g(e) = 0$ pour $e \notin E(G_f)$, ces circuits sont f-augmentants ; au moins l'un d'entre eux a un coût total négatif, ce qui prouve le théorème. □

Ce résultat remonte essentiellement à Tolstoĭ [1930] et a été souvent redécouvert sous des formes différentes. Une formulation équivalente est la suivante :

Corollaire 9.7. (Ford et Fulkerson [1962]) *Soit (G, u, b, c) une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM. *Un b-flot f est de coût minimum si et seulement si (G_f, c) possède un potentiel réalisable.*

Preuve. Par le théorème 9.6 f est un b-flot de coût minimum si et seulement si G_f ne contient pas de circuits négatifs. Par le théorème 7.7 (G_f, c) n'a pas de circuits négatifs si et seulement s'il existe un potentiel réalisable. □

Les potentiels réalisables peuvent être également considérés comme les solutions réalisables du dual du PL associées au PROBLÈME DU FLOT DE COÛT MINIMUM. Cela se démontre par la preuve alternative suivante du critère d'optimalité :
Deuxième démonstration du corollaire 9.7 : écrivons le PROBLÈME DU FLOT DE COÛT MINIMUM comme problème de maximisation et soit le PL suivant

$$\begin{array}{lll} \max & \displaystyle\sum_{e\in E(G)} -c(e)x_e & \\ \text{s.c.} & \displaystyle\sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} x_e = b(v) & (v \in V(G)) \\ & x_e \leq u(e) & (e \in E(G)) \\ & x_e \geq 0 & (e \in E(G)) \end{array} \tag{9.1}$$

et son dual

$$\begin{array}{lll} \min & \displaystyle\sum_{v\in V(G)} b(v)y_v + \sum_{e\in E(G)} u(e)z_e & \\ \text{s.c.} & y_v - y_w + z_e \geq -c(e) & (e = (v,w) \in E(G)) \\ & z_e \geq 0 & (e \in E(G)). \end{array} \tag{9.2}$$

Soit x un b-flot, c.-à-d. une solution réalisable de (9.1). Par le corollaire 3.23 x est optimale si et seulement s'il existe une solution duale réalisable (y, z) de (9.2) telle que x et (y, z) vérifient les conditions des écarts complémentaires

$$z_e(u(e)-x_e) = 0 \text{ et } x_e(c(e)+z_e+y_v-y_w) = 0 \text{ pour tout } e = (v,w) \in E(G).$$

Donc x est optimale si et seulement s'il existe un couple de vecteurs (y, z) tels que

$$\begin{array}{ll} 0 = -z_e \leq c(e) + y_v - y_w & \text{pour } e = (v,w) \in E(G) \text{ avec } x_e < u(e) \quad \text{et} \\ c(e) + y_v - y_w = -z_e \leq 0 & \text{pour } e = (v,w) \in E(G) \text{ avec } x_e > 0. \end{array}$$

Cela est équivalent à l'existence d'un vecteur y tel que $c(e)+y_v-y_w \geq 0$ pour tous les arcs résiduels $e = (v,w) \in E(G_x)$, c.-à-d. à l'existence d'un potentiel réalisable y pour (G_x, c). □

9.3 Algorithme par élimination du circuit moyen minimum

Le théorème de Klein 9.6 suggère l'algorithme suivant : trouver d'abord un b-flot (grâce à un algorithme de flot maximum) et ensuite augmenter successivement ce flot le long des circuits augmentants de coût négatif jusqu'à épuisement de ces circuits. Il nous faut cependant être attentif au choix de ces circuits si nous voulons garantir une borne de complexité polynomiale (voir exercice 7). Une bonne stratégie consiste à choisir un circuit augmentant de coût moyen minimum :

ALGORITHME PAR ÉLIMINATION DU CIRCUIT MOYEN MINIMUM

Input Un graphe orienté G, des capacités $u : E(G) \to \mathbb{R}_+$, des nombres $b : V(G) \to \mathbb{R}$ avec $\sum_{v \in V(G)} b(v) = 0$, et des coûts $c : E(G) \to \mathbb{R}$.

Output Un b-flot de coût minimum f.

① Trouver un b-flot f.

② Trouver un circuit C dans G_f de poids moyen minimum.
If C a un coût total non négatif ou si G_f est sans circuits **then stop**.

③ Calculer $\gamma := \min_{e \in E(C)} u_f(e)$. Augmenter f le long de C par γ.
Go to ②.

Par 9.1, ① peut être implémenté par un algorithme pour le PROBLÈME DU FLOT MAXIMUM. ② peut être implémenté grâce à l'algorithme présenté au paragraphe 7.3. Nous allons maintenant montrer que l'algorithme se termine après un nombre polynomial d'itérations. La preuve sera analogue à celle du paragraphe 8.3. Soit $\mu(f)$ le poids minimum d'un circuit de G_f. Alors un b-flot f est optimum si et seulement si $\mu(f) \geq 0$ par le théorème 9.6.

Montrons que $\mu(f)$ est non décroissant tout au long de l'algorithme. Montrons de plus que $\mu(f)$ s'accroît strictement toutes les $|E(G)|$ itérations. Comme d'habitude n et m sont respectivement le nombre de sommets et d'arcs de G.

Lemme 9.8. *Soit $f_1, f_2, \ldots, f_t$ une suite de b-flots telle que $\mu(f_i) < 0$ et f_{i+1} se déduit de f_i par augmentation le long de C_i, où C_i est un circuit de G_{f_i} de coût moyen minimum, pour $i = 1, \ldots, t-1$. Alors :*

(a) *$\mu(f_k) \leq \mu(f_{k+1})$ pour tout k.*

(b) *$\mu(f_k) \leq \frac{n}{n-2}\,\mu(f_l)$ pour tout $k < l$ tel que $C_k \cup C_l$ contienne une paire d'arcs inversés.*

Preuve. (a) : soient f_k, f_{k+1} deux flots consécutifs de cette suite. Soit H le graphe eulérien résultant de $(V(G), E(C_k) \dot\cup E(C_{k+1}))$ par suppression des paires d'arcs inversés. (Les arcs apparaissant dans C_k et C_{k+1} sont comptés deux fois.) Tout sous-graphe simple de H est un sous-graphe de G_{f_k}, car chaque arc dans $E(G_{f_{k+1}}) \setminus E(G_{f_k})$ doit être l'arc inversé d'un arc dans $E(C_k)$. Puisque H est eulérien, il se décompose en circuits, et chacun de ces circuits a un poids moyen au moins égal à $\mu(f_k)$. Donc $c(E(H)) \geq \mu(f_k)|E(H)|$.

Puisque le poids total de deux arcs inversés est zéro,

$$c(E(H)) \;=\; c(E(C_k)) + c(E(C_{k+1})) \;=\; \mu(f_k)|E(C_k)| + \mu(f_{k+1})|E(C_{k+1})|.$$

Puisque $|E(H)| \leq |E(C_k)| + |E(C_{k+1})|$, nous en concluons que

$$\begin{aligned} \mu(f_k)(|E(C_k)| + |E(C_{k+1})|) &\leq \mu(f_k)|E(H)| \\ &\leq c(E(H)) \\ &= \mu(f_k)|E(C_k)| + \mu(f_{k+1})|E(C_{k+1})|; \end{aligned}$$

cela implique $\mu(f_{k+1}) \geq \mu(f_k)$.

(b) : par (a) il suffit de montrer que (b) est vrai pour les couples k, l tels que pour $k < i < l$, $C_i \cup C_l$ ne contient pas de paires d'arcs inversés.

Comme dans la preuve de (a), soit H le graphe eulérien résultant de $(V(G), E(C_k) \dot\cup E(C_l))$ par suppression des paires d'arcs inversés. H est un sous-graphe de G_{f_k}, car tout arc de $E(C_l) \setminus E(G_{f_k})$ doit être l'arc inversé d'un arc dans un des circuits $C_k, C_{k+1}, \ldots, C_{l-1}$. Mais – par notre choix de k et l – seul C_k parmi ces circuits contient un arc inversé de C_l.

Donc comme dans (a), $c(E(H)) \geq \mu(f_k)|E(H)|$ et

$$c(E(H)) \;=\; \mu(f_k)|E(C_k)| + \mu(f_l)|E(C_l)|.$$

Puisque $|E(H)| \leq |E(C_k)| + \frac{n-2}{n}|E(C_l)|$ (au moins deux arcs ont disparu) nous obtenons

$$\begin{aligned} \mu(f_k)\left(|E(C_k)| + \frac{n-2}{n}\,|E(C_l)|\right) &\leq \mu(f_k)|E(H)| \\ &\leq c(E(H)) \\ &= \mu(f_k)|E(C_k)| + \mu(f_l)|E(C_l)|; \end{aligned}$$

cela implique $\mu(f_k) \leq \frac{n}{n-2}\,\mu(f_l)$. □

Corollaire 9.9. *Durant le déroulement de* L'ALGORITHME PAR ÉLIMINATION DU CIRCUIT MOYEN MINIMUM, *$|\mu(f)|$ décroît au moins de $\frac{1}{2}$ toutes les mn itérations.*

Preuve. Soient $C_k, C_{k+1}, \ldots, C_{k+m}$ les circuits augmentants durant des itérations successives de l'algorithme. Puisque chacun de ces circuits contient un arc seuil (un arc retiré tout de suite après du graphe résiduel), il existe nécessairement deux de ces circuits, disons C_i et C_j ($k \leq i < j \leq k+m$), dont l'union contient une paire d'arcs inversés. Par le lemme 9.8 nous avons

$$\mu(f_k) \;\leq\; \mu(f_i) \;\leq\; \frac{n}{n-2}\,\mu(f_j) \;\leq\; \frac{n}{n-2}\,\mu(f_{k+m}).$$

Donc $|\mu(f)|$ décroît au moins par un facteur $\frac{n-2}{n}$ toutes les m itérations. On en déduit le corollaire puisque $\left(\frac{n-2}{n}\right)^n < e^{-2} < \frac{1}{2}$. □

Cela nous montre que l'algorithme s'exécute en temps polynomial pourvu que les coûts des arcs soient entiers : initialement $|\mu(f)|$ vaut au plus $|c_{\min}|$, $c_{\min}$ étant le coût minimum d'un arc, et diminue d'au moins un facteur $\frac{1}{2}$ toutes les mn itérations. Donc après $O(mn\log(n|c_{\min}|))$ itérations, $\mu(f)$ est plus grand que $-\frac{1}{n}$. Si les coûts des arcs sont entiers, cela signifie que $\mu(f) \geq 0$ et que l'algorithme s'arrête. Donc par le corollaire 7.14, le temps de calcul est $O\left(m^2n^2\log(n|c_{\min}|)\right)$.

Mieux encore, nous pouvons déduire un algorithme fortement polynomial pour le PROBLÈME DU FLOT DE COÛT MINIMUM (obtenu d'abord par Tardos [1985]) :

Théorème 9.10. (Goldberg et Tarjan [1989]) L'ALGORITHME PAR ÉLIMINATION DU CIRCUIT MOYEN MINIMUM *s'exécute en un temps* $O\left(m^3n^2\log n\right)$.

Preuve. Montrons qu'après $mn(\lceil\log n\rceil+1)$ itérations, au moins un arc est fixé, c.-à-d. que le flot ne changera plus jamais sur cet arc. Il y aura donc au plus $O\left(m^2n\log n\right)$ itérations. Le théorème sera prouvé en utilisant le corollaire 8.15 pour ① et le corollaire 7.14 pour ②.

Soit f le flot à une itération, et soit f' le flot $mn(\lceil\log n\rceil+1)$ itérations plus tard. Définissons les poids c' par $c'(e) := c(e) - \mu(f')$ ($e \in E(G_{f'})$). Soit π un potentiel réalisable de $(G_{f'}, c')$ (qui existe par le théorème 7.7). Nous avons $0 \leq c'_\pi(e) = c_\pi(e) - \mu(f')$, donc

$$c_\pi(e) \;\geq\; \mu(f') \quad \text{pour tout } e \in E(G_{f'}). \tag{9.3}$$

Soit C le circuit de poids moyen minimum dans G_f choisi dans l'algorithme pour augmenter f. Puisque, par le corollaire 9.9,

$$\mu(f) \;\leq\; 2^{\lceil\log n\rceil+1}\mu(f') \;\leq\; 2n\mu(f')$$

(voir figure 9.2), nous avons

$$\sum_{e\in E(C)} c_\pi(e) \;=\; \sum_{e\in E(C)} c(e) \;=\; \mu(f)|E(C)| \;\leq\; 2n\mu(f')|E(C)|.$$

Soit donc $e_0 = (x,y) \in E(C)$ avec $c_\pi(e_0) \leq 2n\mu(f')$. Par (9.3) nous avons $e_0 \notin E(G_{f'})$.

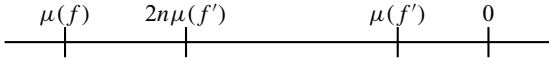


Figure 9.2.

Nous affirmons que, pour tout b-flot f'' tel que $e_0 \in E(G_{f''})$, $\mu(f'') < \mu(f')$.

Par le lemme 9.8(a) cela impliquera que e_0 ne sera plus jamais dans le graphe résiduel, c.-à-d. que e_0 et $\overleftarrow{e_0}$ sont fixés durant $mn(\lceil\log n\rceil+1)$ itérations après que e_0 soit utilisé dans C. Cela terminera la preuve.

Pour montrer cette affirmation, soit f'' un b-flot avec $e_0 \in E(G_{f''})$. En appliquant la proposition 9.4 à f' et f'', nous obtenons une circulation g avec $g(e) = 0$ pour tout $e \notin E(G_{f'})$ et $g(\overleftarrow{e_0}) > 0$ (parce que $e_0 \in E(G_{f''}) \setminus E(G_{f'})$).

Par la proposition 9.5, g peut être écrit comme somme de flots sur des circuits f'-augmentants. Un de ces circuits, disons W, doit contenir $\overleftarrow{e_0}$. En utilisant la relation $c_\pi(\overleftarrow{e_0}) = -c_\pi(e_0) \geq -2n\mu(f')$ et en appliquant (9.3) à chaque $e \in E(W) \setminus \{\overleftarrow{e_0}\}$ nous obtenons une borne inférieure pour le poids total de W :

$$c(E(W)) \;=\; \sum_{e \in E(W)} c_\pi(e) \;\geq\; -2n\mu(f') + (n-1)\mu(f') \;>\; -n\mu(f').$$

Mais le circuit obtenu en inversant les arcs de W est un circuit f''-augmentant (cela se voit en échangeant les rôles de f' et f''), et son poids total est plus petit que $n\mu(f')$. Cela signifie que $G_{f''}$ contient un circuit dont le poids moyen est plus petit que $\mu(f')$; l'affirmation est donc démontrée. □

9.4 Algorithme par plus courts chemins successifs

Le théorème suivant conduit également à un autre algorithme :

Théorème 9.11. (Jewell [1958], Iri [1960], Busacker et Gowen [1961]) *Soit (G, u, b, c) une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM*, et soit f un b-flot de coût minimum. Soit P un plus court chemin de s à t dans G_f (par rapport à c) (pour un couple s, t). Soit f' le flot obtenu en augmentant f le long de P par au plus la capacité résiduelle de P. Alors f' est un b'-flot de coût minimum.*

Preuve. f' est un b'-flot pour un certain b'. Supposons que f' ne soit pas de coût minimum. Par le théorème 9.6, il existe un circuit C dans $G_{f'}$ de coût négatif. Soit H le graphe résultant de $(V(G), E(C) \dot\cup E(P))$ en supprimant les paires d'arcs inversés (de nouveau, les arcs appartenant à C et P sont comptés deux fois).

Si $e \in E(G_{f'}) \setminus E(G_f)$, l'arc inversé de e est dans $E(P)$ et $E(H) \subseteq E(G_f)$.

$c(E(H)) = c(E(C)) + c(E(P)) < c(E(P))$. De plus, H est l'union d'un chemin de s à t et de circuits. Mais puisque $E(H) \subseteq E(G_f)$, aucun des circuits n'a un poids négatif (sinon f ne serait pas un b-flot de coût minimum).

Donc H, et aussi G_f, contiennent un chemin de s à t de poids plus petit que celui de P, ce qui contredit notre choix de P. □

Si les poids sont conservatifs, nous pouvons commencer avec une circulation $f \equiv 0$ comme circulation optimale (b-flot avec $b \equiv 0$). Sinon nous pouvons saturer initialement tous les arcs de coût négatif et de capacité bornée. Cela modifie les valeurs b, mais nous assure qu'il n'y a pas de circuit augmentant négatif (c.-à-d. que c est conservatif pour G_f) sauf si l'instance est non bornée.

ALGORITHME PAR PLUS COURTS CHEMINS SUCCESSIFS

Input Un graphe orienté G, des capacités $u : E(G) \to \mathbb{R}_+$, des nombres $b : V(G) \to \mathbb{R}$ avec $\sum_{v \in V(G)} b(v) = 0$, et des poids conservatifs $c : E(G) \to \mathbb{R}$.

Output Un b-flot de coût minimum f.

① $b' := b$ et $f(e) := 0$ pour tout $e \in E(G)$.

② **If** $b' = 0$ **then stop**, **else** :
Choisir un sommet s avec $b'(s) > 0$.
Choisir un sommet t avec $b'(t) < 0$ et t connecté à s dans G_f.
If aucun t n'existe **then stop**. (Il n'y a pas de b-flot.)

③ Trouver un plus court chemin P de s à t dans G_f.

④ Calculer $\gamma := \min \left\{ \min_{e \in E(P)} u_f(e), b'(s), -b'(t) \right\}$.
$b'(s) := b'(s) - \gamma$ et $b'(t) := b'(t) + \gamma$. Augmenter f le long de P par γ.
Go to ②.

Si les capacités sont arbitraires, nous aurons les mêmes difficultés que dans l'ALGORITHME DE FORD-FULKERSON (voir exercice 2 du chapitre 8 en mettant tous les coûts à zéro). Nous supposerons donc que les quantités u et b sont entières. Il est alors évident que l'algorithme se termine après $B := \frac{1}{2} \sum_{v \in V(G)} |b(v)|$ augmentations. Par le théorème 9.11, le flot résultant est optimum si le flot initial est optimum. Cela est vrai si et seulement si les coûts c sont conservatifs.

Remarquons que si l'algorithme décide qu'il n'y a pas de b-flot, cette réponse est correcte. C'est une observation facile à faire ; voir l'exercice 13.

Chaque augmentation nécessite un calcul de plus court chemin. Puisqu'il y a des coûts négatifs, il nous faut utiliser l'ALGORITHME DE MOORE-BELLMAN-FORD ayant un temps de calcul $O(nm)$ (théorème 7.5) : ainsi le temps total de calcul sera $O(Bnm)$. Cependant, comme dans la preuve du théorème 7.9, on peut se ramener (sauf au début) à une situation où tous les coûts sont non négatifs :

Théorème 9.12. (Tomizawa [1971], Edmonds et Karp [1972]) *Si u et b sont entiers, l'*ALGORITHME PAR PLUS COURTS CHEMINS SUCCESSIFS *peut s'exécuter en un temps $O\left(nm + B(m + n \log n)\right)$, où $B = \frac{1}{2} \sum_{v \in V(G)} |b(v)|$.*

Preuve. On peut supposer qu'il n'y a qu'une source s ; sinon introduisons un nouveau sommet s, les arcs (s, v) avec capacité $\max\{0, b(v)\}$ et des coûts nuls pour tout $v \in V(G)$. Puis posons $b(s) := B$ et $b(v) := 0$ pour chaque source originale v. Nous obtenons ainsi un problème équivalent avec une source. De plus, nous pouvons supposer que tous les sommets sont connectés à s (les autres sommets peuvent être éliminés).

Introduisons un potentiel $\pi_i : V(G) \to \mathbb{R}$ à chaque itération i de l'ALGORITHME PAR PLUS COURTS CHEMINS SUCCESSIFS. Commençons avec un potentiel réalisa-

ble π_0 de (G, c). Par le corollaire 7.8, ce potentiel existe et se calcule en un temps $O(mn)$.

Supposons maintenant que f_{i-1} soit le flot avant l'itération i. Le calcul du plus court chemin à l'itération i s'effectue avec les coûts réduits $c_{\pi_{i-1}}$ au lieu de c. Soit $l_i(v)$ la longueur d'un plus court chemin de s à v dans $G_{f_{i-1}}$ avec les poids $c_{\pi_{i-1}}$. Posons alors $\pi_i(v) := \pi_{i-1}(v) + l_i(v)$.

Montrons par induction sur i que π_i est un potentiel réalisable pour (G_{f_i}, c). Cela est évident pour $i = 0$. Pour $i > 0$ et un arc quelconque $e = (x, y) \in E(G_{f_{i-1}})$ nous avons (par définition de l_i et l'hypothèse d'induction)

$$l_i(y) \;\leq\; l_i(x) + c_{\pi_{i-1}}(e) \;=\; l_i(x) + c(e) + \pi_{i-1}(x) - \pi_{i-1}(y),$$

donc

$$c_{\pi_i}(e) \;=\; c(e) + \pi_i(x) - \pi_i(y) \;=\; c(e) + \pi_{i-1}(x) + l_i(x) - \pi_{i-1}(y) - l_i(y) \;\geq\; 0.$$

Pour tout $e = (x, y) \in P_i$ (où P_i est le chemin augmentant à l'itération i) nous avons

$$l_i(y) \;=\; l_i(x) + c_{\pi_{i-1}}(e) \;=\; l_i(x) + c(e) + \pi_{i-1}(x) - \pi_{i-1}(y),$$

donc $c_{\pi_i}(e) = 0$, et l'arc inversé de e a aussi un poids nul. Puisque chaque arc de $E(G_{f_i}) \setminus E(G_{f_{i-1}})$ est l'arc inversé d'un arc dans P_i, c_{π_i} est bien une fonction non négative sur $E(G_{f_i})$.

Observons que, pour tout i et quel que soit t, les plus courts chemins de s à t par rapport à c sont aussi les plus courts chemins de s à t par rapport à c_{π_i}, puisque $c_{\pi_i}(P) - c(P) = \pi_i(s) - \pi_i(t)$ pour tout chemin P de s à t.

Nous utiliserons donc l'ALGORITHME DE DIJKSTRA – qui a une complexité $O(m + n \log n)$ quand il est implémenté avec un tas de Fibonacci par le théorème 7.4 – pour tous les calculs de plus courts chemins sauf pour le premier. Puisqu'il y a au plus B itérations, le temps total de calcul est $O(nm + B(m + n \log n))$. □

Notons que (en contraste avec de nombreux autres problèmes, par exemple le PROBLÈME DU FLOT MAXIMUM) nous ne pouvons pas supposer que le graphe est simple quand nous étudions le PROBLÈME DU FLOT DE COÛT MINIMUM. Le temps de calcul du théorème 9.12 est exponentiel sauf si B est petit. Si $B = O(n)$, cet algorithme est l'algorithme le plus rapide connu (voir paragraphe 11.1 pour une application).

Dans le reste de ce paragraphe nous allons voir comment modifier cet algorithme afin de réduire le nombre de calculs de plus courts chemins. Nous étudierons seulement le cas où les capacités sont infinies. Par le lemme 9.2, on peut transformer chaque instance du PROBLÈME DU FLOT DE COÛT MINIMUM en une instance équivalente avec des capacités infinies.

L'idée de base – d'Edmonds et Karp [1972] – est la suivante. Durant les premières itérations nous considérons les chemins augmentants pour lesquels γ – la quantité de flot qui peut être envoyée – est grande. Nous commençons avec

$\gamma = 2^{\lfloor \log b_{\max} \rfloor}$ puis nous réduirons γ par un facteur de deux si aucune nouvelle augmentation γ ne peut se faire. Après $\lfloor \log b_{\max} \rfloor + 1$ itérations $\gamma = 1$ et nous arrêtons (nous pouvons supposer b entier). Cette technique de réduction d'échelle a montré son utilité dans de nombreux algorithmes (voir aussi exercice 14). Donnons une description du premier algorithme de réduction d'échelle :

ALGORITHME PAR RÉDUCTION D'ÉCHELLE DES CAPACITÉS

Input Un graphe orienté G avec des capacités infinies $u(e) = \infty$ ($e \in E(G)$), des nombres $b : V(G) \to \mathbb{Z}$ avec $\sum_{v \in V(G)} b(v) = 0$, et des poids conservatifs $c : E(G) \to \mathbb{R}$.

Output un b-flot de coût minimum f.

① Poser $b' := b$ et $f(e) := 0$ pour tout $e \in E(G)$.
Poser $\gamma = 2^{\lfloor \log b_{\max} \rfloor}$, où $b_{\max} = \max\{b(v) : v \in V(G)\}$.

② **If** $b' = 0$ **then stop**, **else** :
Choisir un sommet s avec $b'(s) \geq \gamma$.
Choisir un sommet t avec $b'(t) \leq -\gamma$ tel que t soit connecté à s dans G_f.
If un tel s ou t n'existent pas **then go to** ⑤.

③ Trouver un plus court chemin P de s à t dans G_f.

④ $b'(s) := b'(s) - \gamma$ et $b'(t) := b'(t) + \gamma$. Augmenter f le long de P par γ.
Go to ②.

⑤ **If** $\gamma = 1$ **then stop**. (Il n'existe pas de b-flot.)
Else $\gamma := \frac{\gamma}{2}$ et **go to** ②.

Théorème 9.13. (Edmonds et Karp [1972]) *L'*ALGORITHME PAR RÉDUCTION D'ÉCHELLE DES CAPACITÉS *résout de manière correcte le* PROBLÈME DU FLOT DE COÛT MINIMUM *pour b entier, des capacités infinies et des poids conservatifs. Son temps de calcul est $O(n(m + n \log n) \log b_{\max})$, où $b_{\max} = \max\{b(v) : v \in V(G)\}$.*

Preuve. Comme précédemment, l'algorithme répond correctement par le théorème 9.11. Notons qu'à chaque instant, la capacité résiduelle d'un arc est soit infinie, soit un multiple entier de γ.

Pour trouver la borne de complexité, appelons *phase* la période durant laquelle γ reste constant. Montrons qu'il y a moins de $4n$ augmentations dans chaque phase. Supposons que cela soit faux. Soient f le flot au début et g le flot à la fin d'une phase de valeur γ. $g - f$ peut être considéré comme un b''-flot dans G_f, où $\sum_{x \in V(G)} |b''(x)| \geq 8n\gamma$. Soit $S := \{x \in V(G) : b''(x) > 0\}$, $S^+ := \{x \in V(G) : b''(x) \geq 2\gamma\}$, $T := \{x \in V(G) : b''(x) < 0\}$, $T^+ := \{x \in V(G) : b''(x) \leq -2\gamma\}$. S'il avait existé un chemin de S^+ à T^+ dans G_f, la phase de valeur 2γ se serait poursuivie. Donc le total des valeurs b'' pour toutes les sources connectées à S^+ dans G_f est plus grand que $n(-2\gamma)$. Donc $\sum_{x \in S^+} b''(x) < 2n\gamma$ (notons qu'il existe un b''-flot dans G_f). Nous avons alors

$$\sum_{x\in V(G)} |b''(x)| \;=\; 2\sum_{x\in S} b''(x) \;=\; 2\left(\sum_{x\in S^+} b''(x) + \sum_{x\in S\setminus S^+} b''(x)\right)$$
$$< \;\; 2(2n\gamma + 2n\gamma) \;=\; 8n\gamma,$$

ce qui mène à une contradiction.

Cela signifie que le nombre total de calculs de plus courts chemins est de l'ordre de $O(n \log b_{\max})$. Combinant ce résultat avec la technique du théorème 9.12, nous obtenons la borne $O(mn + n \log b_{\max}(m + n \log n))$. □

Cet algorithme a été le premier algorithme polynomial pour le PROBLÈME DU FLOT DE COÛT MINIMUM. Avec d'autres modifications, il est possible d'obtenir un algorithme fortement polynomial. Ce sera l'objet du paragraphe suivant.

9.5 Algorithme d'Orlin

L'ALGORITHME PAR RÉDUCTION D'ÉCHELLE DES CAPACITÉS du paragraphe précédent peut être amélioré. En effet, si à une étape de l'ALGORITHME PAR RÉDUCTION D'ÉCHELLE DES CAPACITÉS le flot sur un arc dépasse $8n\gamma$, cet arc pourra être contracté. Observons en effet que le flot sur cet arc restera toujours positif (et que par conséquent son coût réduit par rapport à tout potentiel réalisable sera zéro dans le graphe résiduel) : il y a au plus $4n$ augmentations dans une phase de valeur γ, $4n$ augmentations avec $\frac{\gamma}{2}$, et ainsi de suite ; donc la valeur totale ajoutée ou retranchée sur cet arc ne peut excéder $8n\gamma$.

Nous allons décrire l'ALGORITHME D'ORLIN sans utiliser la contraction. Cela simplifiera la présentation, en particulier dans la perspective d'une implémentation. Un ensemble F gardera trace des arcs (et de leurs arcs inversés) que l'on peut contracter. Pour chaque composante connexe de $(V(G), F)$ un seul sommet y aura une valeur $b(y)$ non nulle ; y sera le représentant de la composante connexe ; si x est un sommet de cette composante connexe, nous poserons $r(x) = y$.

L'ALGORITHME D'ORLIN ne demande pas que b soit entier, mais il suppose que les capacités sont infinies (cela n'est pas une restriction par le lemme 9.2).

ALGORITHME D'ORLIN

Input Un graphe orienté G avec des capacités infinies $u(e) = \infty$ ($e \in E(G)$), des nombres $b : V(G) \to \mathbb{R}$ avec $\sum_{v\in V(G)} b(v) = 0$, et des poids conservatifs $c : E(G) \to \mathbb{R}$.

Output Un b-flot de coût minimum f.

① Poser $b' := b$ et $f(e) := 0$ pour tout $e \in E(G)$.
Poser $r(v) := v$ pour tout $v \in V(G)$. Poser $F := \emptyset$.
Calculer $\gamma = \max_{v\in V(G)} |b'(v)|$.

② **If** $b' = 0$ **then stop**.

③ Choisir un sommet s tel que $b'(s) > \frac{n-1}{n}\gamma$.
If un tel s n'existe pas **then go to** ④.
Choisir un sommet t avec $b'(t) < -\frac{1}{n}\gamma$ et t connecté à s dans G_f.
If un tel sommet t n'existe pas **then stop**. (Il n'existe pas de b-flot.)
Go to ⑤.

④ Choisir un sommet t avec $b'(t) < -\frac{n-1}{n}\gamma$.
If un tel sommet t n'existe pas **then go to** ⑥.
Choisir un sommet s avec $b'(s) > \frac{1}{n}\gamma$ tel que t soit connecté à s dans G_f.
If un tel s n'existe pas **then stop**. (Il n'existe pas de b-flot.)

⑤ Trouver un plus court chemin P de s à t dans G_f.
$b'(s) := b'(s) - \gamma$ et $b'(t) := b'(t) + \gamma$. Augmenter f le long de P par γ.
Go to ②.

⑥ **If** $f(e) = 0$ pour tout $e \in E(G) \setminus F$ **then** $\gamma := \min\left\{\frac{\gamma}{2}, \max_{v \in V(G)} |b'(v)|\right\}$,
else $\gamma := \frac{\gamma}{2}$.

⑦ **For** tout $e = (x, y) \in E(G) \setminus F$ avec $r(x) \neq r(y)$ et $f(e) > 8n\gamma$ **do** :
$F := F \cup \{e, \overleftarrow{e}\}$.
Soit $x' := r(x)$ et $y' := r(y)$. Soit Q le chemin de x' à y' dans F.
If $b'(x') > 0$ **then** augmenter f le long de Q par $b'(x')$,
else augmenter f le long du chemin inversé de Q par $-b'(x')$.
$b'(y') := b'(y') + b'(x')$ et $b'(x') := 0$.
$r(z) := y'$ pour tous les z connectés à y' dans F.

⑧ **Go to** ②.

Cet algorithme a été proposé par Orlin [1993] ; voir également Plotkin et Tardos [1990]. Montrons d'abord qu'il s'exécute correctement. Appelons **phase** le temps entre deux changements de γ.

Lemme 9.14. L'ALGORITHME D'ORLIN *résout correctement le* PROBLÈME DU FLOT DE COÛT MINIMUM *avec des coûts conservatifs. À chaque étape f est un $(b - b')$-flot de coût minimum.*

Preuve. Montrons que f est toujours un $(b - b')$-flot et qu'en particulier f est toujours non négatif. Observons qu'à chaque instant la capacité résiduelle d'un arc qui n'est pas dans F est infinie ou est un multiple entier de γ. D'autre part, tout arc $e \in F$ a toujours une capacité résiduelle positive : en effet, il y a dans chaque phase au plus $n - 1$ augmentations inférieures à $2\frac{n-1}{n}\gamma$ dans ⑦ et au plus $2n$ augmentations par γ dans ⑤ ; donc la quantité totale de flot déplacée après que e soit devenu un élément de F dans la phase de valeur γ est inférieure à $8n\gamma$.

f est toujours non négatif et est donc un $(b - b')$-flot. Nous affirmons que f est également un $(b - b')$-flot de coût minimum et que chaque chemin de v à w dans F est un plus court chemin de v à w dans G_f. La première de ces conditions implique

en fait la seconde puisque, par le théorème 9.6, il n'existe pas de circuit négatif dans G_f quand f est un flot de coût minimum. Notre affirmation se déduit du théorème 9.11 : P dans ⑤ et Q dans ⑦ sont tous deux des plus courts chemins.

Montrons enfin que si l'algorithme stoppe en ③ ou en ④ avec $b' \neq 0$, alors il n'existe pas de b-flot. Supposons que l'algorithme s'arrête en ③, ce qui implique qu'un sommet s avec $b'(s) > \frac{n-1}{n}\gamma$ existe, mais que aucun sommet t avec $b'(t) < -\frac{1}{n}\gamma$ n'est connecté à s dans G_f. Soit alors R l'ensemble des sommets connectés à s dans G_f. Puisque f est un $(b-b')$-flot, $\sum_{x\in R}(b(x)-b'(x)) = 0$. Donc

$$\sum_{x\in R} b(x) \;=\; \sum_{x\in R}(b(x)-b'(x)) + \sum_{x\in R} b'(x) \;=\; \sum_{x\in R} b'(x) \;=\; b'(s) + \sum_{x\in R\setminus\{s\}} b'(x) \;>\; 0.$$

Cela montre qu'aucun b-flot n'existe. On a une preuve analogue quand l'algorithme termine en ④. □

Analysons maintenant le temps de calcul.

Lemme 9.15. (Plotkin et Tardos [1990]) *Si $|b'(s)| > \frac{n-1}{n}\gamma$ pour un sommet s à une étape de l'algorithme, la composante connexe de $(V(G), F)$ contenant ce sommet augmente durant les $\lceil 2\log n + \log m\rceil + 4$ phases suivantes.*

Preuve. Soit $|b'(s)| > \frac{n-1}{n}\gamma_1$ pour un sommet s au début d'une phase de l'algorithme avec $\gamma = \gamma_1$. Soit γ_0 la valeur de la phase précédente et γ_2 la valeur de la phase $\lceil 2\log n + \log m\rceil + 4$ phases plus tard. $\frac{1}{2}\gamma_0 \geq \gamma_1 \geq 16n^2 m\gamma_2$. Soient b'_1 et f_1 les quantités b' et f au début de la phase de valeur γ_1 et b'_2 et f_2 les quantités b' et f à la fin de la phase de valeur γ_2.

Soit S la composante connexe de $(V(G), F)$ contenant s dans la phase de valeur γ_1, et supposons que cette composante reste inchangée durant les $\lceil 2\log n + \log m\rceil + 4$ phases suivantes. ⑦ assure que $b'(v) = 0$ pour tous les v avec $r(v) \neq v$. Donc $b'(v) = 0$ pour tout $v \in S \setminus \{s\}$ et

$$\sum_{x\in S} b(x) - b'_1(s) \;=\; \sum_{x\in S}(b(x) - b'_1(x)) \;=\; \sum_{e\in\delta^+(S)} f_1(e) - \sum_{e\in\delta^-(S)} f_1(e). \quad (9.4)$$

Nous affirmons que

$$\left|\sum_{x\in S} b(x)\right| \;\geq\; \frac{1}{n}\gamma_1. \quad (9.5)$$

Si $\gamma_1 < \frac{\gamma_0}{2}$, alors chaque arc non dans F a un flot zéro ; donc le membre de droite de l'équation (9.4) vaut zéro, ce qui implique $\left|\sum_{x\in S} b(x)\right| = |b'_1(s)| > \frac{n-1}{n}\gamma_1 \geq \frac{1}{n}\gamma_1$.

Dans l'autre cas ($\gamma_1 = \frac{\gamma_0}{2}$) nous avons

$$\frac{1}{n}\gamma_1 \;\leq\; \frac{n-1}{n}\gamma_1 \;<\; |b'_1(s)| \;\leq\; \frac{n-1}{n}\gamma_0 \;=\; \gamma_0 - \frac{2}{n}\gamma_1. \quad (9.6)$$

Puisque le flot de chaque arc non dans F est un multiple entier de γ_0, le membre de droite de l'équation (9.4) est aussi un multiple entier de γ_0. Cela combiné avec (9.6) implique (9.5).

Considérons alors le f_2-flot sur tous les arcs sortant de S moins le flot total sur tous les arcs entrant dans S. Puisque f_2 est un $(b - b'_2)$-flot, cette quantité vaut $\sum_{x\in S} b(x) - b'_2(s)$. Utilisant (9.5) et $|b'_2(s)| \leq \frac{n-1}{n}\gamma_2$, nous obtenons

$$\sum_{e\in\delta^+(S)\cup\delta^-(S)} |f_2(e)| \;\geq\; \left|\sum_{x\in S} b(x)\right| - |b'_2(s)| \;\geq\; \frac{1}{n}\gamma_1 - \frac{n-1}{n}\gamma_2$$
$$\geq\; (16nm-1)\gamma_2 \;>\; m(8n\gamma_2).$$

Donc il existe au moins un arc e ayant exactement une extrémité dans S et tel que $f_2(e) > 8n\gamma_2$. Par ⑦ de l'algorithme, la taille de l'ensemble S augmente. □

Théorème 9.16. (Orlin [1993]) L'ALGORITHME D'ORLIN *résout correctement* LE PROBLÈME DU FLOT DE COÛT MINIMUM *avec des poids conservatifs en un temps* $O(n \log m\ (m + n \log n))$.

Preuve. L'algorithme s'exécute correctement par le lemme 9.14. ⑦ requiert un temps $O(mn)$. Par le lemme 9.15, le nombre total de phases est $O(n \log m)$. Ce lemme dit aussi : si s est un sommet et $S \subseteq V(G)$ il y a au plus $\lceil 2\log n + \log m\rceil + 4$ augmentations dans ⑤ démarrant en s si S est la composante connexe de $(V(G), F)$ contenant s. Puisque tous les sommets v tels que $r(v) \neq v$ vérifient $b'(v) = 0$ à chaque instant, il y a au plus $\lceil 2\log n + \log m\rceil + 4$ accroissements pour chaque ensemble S qui est, à une étape de l'algorithme, une composante connexe de F. Puisque la famille de ces ensembles est laminaire, il y a au plus $2n - 1$ ensembles de ce type (corollaire 2.15) et $O(n \log m)$ augmentations dans ⑤.

Utilisant la technique développée au théorème 9.12, nous obtenons un temps d'exécution $O\left(mn + (n \log m)(m + n \log n)\right)$. □

Cela est la meilleure borne de complexité connue pour le PROBLÈME DU FLOT DE COÛT MINIMUM sans capacités.

Théorème 9.17. (Orlin [1993]) *Le* PROBLÈME DU FLOT DE COÛT MINIMUM *avec* $n = |V(G)|$ *et* $m = |E(G)|$ *peut se résoudre en* $O\left(m \log m(m + n \log n)\right)$.

Preuve. Nous utiliserons la construction du lemme 9.2. Nous avons à résoudre un PROBLÈME DU FLOT DE COÛT MINIMUM sans capacités sur un graphe biparti H : $V(H) = A' \dot\cup B'$, avec $A' = E(G)$ et $B' = V(G)$. Puisque H est sans circuit, un potentiel réalisable initial peut être trouvé en un temps $O(|E(H)|) = O(m)$. Le temps total de calcul est borné par $O(m \log m)$ calculs de plus courts chemins dans un sous-graphe de $\overleftrightarrow{H}$ ayant des coûts non négatifs (voir théorème 9.16).

Avant d'utiliser l'ALGORITHME DE DIJKSTRA, effectuons sur chaque sommet $a \in A'$ qui n'est pas une des extrémités d'un des chemins que nous recherchons la modification suivante : ajouter un arc (b, b') pour chaque paire $(b, a), (a, b')$ et attribuer à cet arc le poids somme des poids de (b, a) et (a, b') ; finalement supprimer a. L'instance du PROBLÈME DE PLUS COURT CHEMIN ainsi obtenue est manifestement équivalente. Puisque chaque sommet de A' a quatre arcs incidents dans $\overleftrightarrow{H}$, le

graphe résultant a $O(m)$ arcs et au plus $n+2$ sommets. Ce prétraitement s'effectue en un temps constant par sommet, c.-à-d. $O(m)$. Cela est également vrai pour le calcul final des chemins dans $\overleftrightarrow{H}$ et des labels de distance pour les sommets effacés. Nous avons donc comme temps total de calcul $O\left((m \log m)(m + n \log n)\right)$. □

Cela est le plus rapide des algorithmes pour le PROBLÈME DU FLOT DE COÛT MINIMUM. Un algorithme de même complexité, mais qui considère directement des instances avec capacités a été proposé par Vygen [2002].

9.6 Algorithme network simplex

Le PROBLÈME DU FLOT DE COÛT MINIMUM peut se formuler comme un PROBLÈME DE PROGRAMMATION LINÉAIRE. En appliquant l'ALGORITHME DU SIMPLEXE et en exploitant la structure particulière du PL, nous obtiendrons un algorithme connu sous le nom d'ALGORITHME NETWORK SIMPLEX. Commençons par caractériser les solutions de base du PL (bien que cela ne soit pas nécessaire pour montrer que l'algorithme s'exécute correctement).

Définition 9.18. *Soit (G, u, b, c) une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM. *Un b-flot f dans (G, u) est un* **flot à support minimal** *si $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$ ne contient pas de cycle.*

Proposition 9.19. *Une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM *a soit une solution optimale qui est un flot à support minimal, soit n'a pas de solution.*

Preuve. Soit f une solution optimale et soit C un cycle de $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$. Il existe dans G_f deux circuits C' et C'' dans G_f associés à C. Soit ϵ la capacité résiduelle minimum dans $E(C') \cup E(C'')$. Nous obtenons deux solutions réalisables f', f'' en augmentant f par ϵ le long respectivement de C' et C''. Comme $2c(f) = c(f') + c(f'')$, f' et f'' sont aussi solutions optimales. Au moins un de ces deux flots a moins d'arcs e vérifiant $0 < f(e) < u(e)$ que f ; donc, après au plus $|E(G)|$ itérations nous trouvons une solution optimale qui est un flot à support minimal. □

Corollaire 9.20. *Soit (G, u, b, c) une instance du* PROBLÈME DE COÛT MINIMUM. *Les solutions de base de*

$$\left\{ x \in \mathbb{R}^{E(G)} \;:\; 0 \leq x_e \leq u(e)\ (e \in E(G)),\right.$$
$$\left. \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b(v)\ (v \in V(G)) \right\}$$

sont les flots à support minimal de (G, u, b, c).

Preuve. La proposition 9.19 montre que toute solution réalisable est un flot à support minimal.

Si f est un flot à support minimal, considérons les inégalités $x_e \geq 0$ pour $e \in E(G)$ et $f(e) = 0$, $x_e \leq u(e)$ pour $e \in E(G)$ et $f(e) = u(e)$, et $\sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} x_e = b(v)$ pour tout v à l'exception d'un seul sommet de chaque composante connexe de $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$. Ces $|E(G)|$ inégalités sont toutes satisfaites par f avec égalité, et la sous-matrice associée à ces contraintes est non singulière. f est donc une solution de base. □

Il y a trois types d'arcs dans un flot à support minimal : ceux de flot nul, ceux avec des capacités saturées et ceux sur lesquels le flot est positif sans atteindre la capacité. Ce dernier ensemble ne contient pas de cycle, c'est donc une forêt orientée, à laquelle on peut ajouter des arcs pour former un «arbre couvrant orienté» (arbre couvrant dans le graphe non orienté associé) quand G est connexe.

Définition 9.21. *Soit (G, u, b, c) une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM *avec G connexe. Une* **structure d'arbre couvrant** *est un quadruplet (r, T, L, U) où $r \in V(G)$, $E(G) = T \dot\cup L \dot\cup U$, $|T| = |V(G)| - 1$, et $(V(G), T)$ ne contient pas de cycles.*

Le b-flot **associé à la structure d'arbre couvrant** *(r, T, L, U) est défini par :*

- *$f(e) := 0$ pour $e \in L$;*
- *$f(e) := u(e)$ pour $e \in U$;*
- *$f(e) := \sum_{v\in C_e} b(v) + \sum_{e\in U\cap\delta^-(C_e)} u(e) - \sum_{e\in U\cap\delta^+(C_e)} u(e)$,*
  C_e étant la composante connexe de $(V(G), T \setminus \{e\})$ contenant v, si $e = (v, w)$.

(r, T, L, U) sera dit **réalisable** *si $0 \leq f(e) \leq u(e)$ pour tout $e \in T$.*

Nous dirons qu'un arc (v, w) de T est descendant si v appartient à la chaîne de r à w dans T ; sinon nous dirons qu'il est montant. (r, T, L, U) sera **fortement réalisable** *si $0 < f(e) \leq u(e)$ pour chaque arc descendant $e \in T$ et $0 \leq f(e) < u(e)$ pour chaque arc montant $e \in T$.*

L'unique fonction $\pi : V(G) \to \mathbb{R}$ avec $\pi(r) = 0$ et $c_\pi(e) = 0$ pour tout $e \in T$ sera appelée **potentiel associé à la structure d'arbre couvrant** *(r, T, L, U).*

Le b-flot f associé à une structure d'arbre couvrant vérifie $\sum_{e\in\delta^+(v)} f(e) - \sum_{e\in\delta^-(v)} f(e) = b(v)$ pour tout $v \in V(G)$ (bien que ce ne soit pas toujours un b-flot réalisable). De plus, nous avons :

Proposition 9.22. *Soit une instance (G, u, b, c) du* PROBLÈME DU FLOT DE COÛT MINIMUM *et une structure d'arbre couvrant (r, T, L, U) ; le b-flot f et le potentiel π associés à cette structure peuvent être respectivement calculés en un temps $O(m)$ et $O(n)$.*

De plus, f est entier si les quantités b et u sont entières, et π est entier si c est entier.

Preuve. Le potentiel associé à (r, T, L, U) se calcule par l'ALGORITHME DE BALAYAGE DE GRAPHES appliqué aux arcs de T et à leurs arcs inversés. Le b-flot

associé à (r, T, L, U) se calcule en temps linéaire, en balayant les sommets dans un ordre de distance non croissante par rapport à r. Les propriétés d'intégralité se déduisent immédiatement de la définition. □

L'ALGORITHME NETWORK SIMPLEX met à jour une structure d'arbre couvrant fortement réalisable qu'il améliore jusqu'à l'optimum. Le critère d'optimalité du corollaire 9.7 implique immédiatement :

Proposition 9.23. *Soit (r, T, L, U) une structure d'arbre couvrant réalisable et son potentiel associé π. Supposons que :*

- $c_\pi(e) \geq 0$ *pour tout* $e \in L$, *et*
- $c_\pi(e) \leq 0$ *pour tout* $e \in U$.

Alors (r, T, L, U) correspond à un b-flot optimal. □

Notons que $\pi(v)$ est le coût du chemin de r à v dans $\overleftrightarrow{G}$ ne contenant que des arcs de T ou des arcs inversés d'arcs de T. Si $e = (v, w) \in E(\overleftrightarrow{G})$ définissons le *circuit fondamental* de e comme l'ensemble constitué de e et des arcs du chemin de w à v incluse dans T ainsi que leurs arcs inversés. Le sommet de C qui est le plus proche de r dans T est appelé le *pic* de C.

Si $e = (v, w) \notin T$, $c_\pi(e) = c(e) + \pi(v) - \pi(w)$ est le coût d'une unité supplémentaire de flot circulant le long du circuit fondamental de e.

Il y a plusieurs manières pour obtenir une structure d'arbre couvrant initiale fortement réalisable. Par exemple, on peut calculer un b-flot (en résolvant un PROBLÈME DU FLOT MAXIMUM), puis appliquer la procédure de la preuve de la proposition 9.19, choisir r arbitrairement, et définir T, L, U suivant les valeurs du flot sur les arcs (en ajoutant si nécessaire des arcs supplémentaires à T). On peut également utiliser la «phase un» de la MÉTHODE DU SIMPLEXE.

Cependant, la méthode la plus simple consiste à ajouter des arcs auxiliaires, ayant des coûts très grands et des grandes capacités, entre r et chaque autre sommet : pour chaque puits $v \in V(G) \setminus \{r\}$ nous introduisons un arc (r, v) avec une capacité $-b(v)$, et pour tout autre sommet $v \in V(G) \setminus \{r\}$ nous introduisons un arc (v, r) de capacité $b(v) + 1$. Le coût de chaque arc auxiliaire doit être suffisamment important pour que ces arcs n'interviennent dans la solution optimale, par exemple $1 + (|V(G)| - 1) \max_{e \in E(G)} |c(e)|$ (exercice 19). Nous pouvons alors choisir pour T l'ensemble des arcs auxiliaires, pour L l'ensemble des arcs originaux, et choisir $U := \emptyset$, afin d'avoir une structure d'arbre couvrant fortement réalisable.

ALGORITHME NETWORK SIMPLEX	
Input	Une instance (G, u, b, c) du PROBLÈME DU FLOT DE COÛT MINIMUM et une structure d'arbre couvrant minimum forte (r, T, L, U).
Output	Une solution optimale f.

① Calculer le b-flot f et le potentiel π associé à (r, T, L, U).

② Soit $e \in L$ avec $c_\pi(e) < 0$ ou $e \in U$ avec $c_\pi(e) > 0$.
If un tel e n'existe pas **then stop**.

③ Soit C le circuit fondamental de e (si $e \in L$) ou de $\overleftarrow{e}$ (si $e \in U$).
Soit $\gamma := c_\pi(e)$.

④ Soit $\delta := \min_{e' \in E(C)} u_f(e')$, et soit e' le dernier arc où le minimum est atteint quand on parcourt C dans le sens de son orientation, en partant de son pic.
Soit $e_0 \in E(G)$ tel que e' soit égal à e_0 ou $\overleftarrow{e_0}$.

⑤ Retirer e de L ou U.
$T := (T \cup \{e\}) \setminus \{e_0\}$.
If $e' = e_0$ **then** insérer e_0 dans U **else** insérer e_0 dans L.

⑥ Augmenter f par δ le long de C.
Soit X la composante connexe de $(V(G), T \setminus \{e\})$ contenant r.
If $e \in \delta^+(X)$ **then** $\pi(v) := \pi(v) + \gamma$ pour $v \in V(G) \setminus X$.
If $e \in \delta^-(X)$ **then** $\pi(v) := \pi(v) - \gamma$ pour $v \in V(G) \setminus X$.
Go to ②.

Remarquons que l'étape ⑥ pourrait simplement être simplifiée par l'instruction «aller à ①», puisque f et π calculés en ⑥ sont associés à la nouvelle structure d'arbre couvrant. Notons aussi qu'il est possible que $e = e_0$; dans ce cas nous aurons : $X = V(G)$, T, f et π ne changent pas, mais e est transféré de L à U ou le contraire.

Théorème 9.24. (Dantzig [1951], Cunningham [1976]) *L'*ALGORITHME NETWORK SIMPLEX *s'arrête après un nombre fini d'itérations et retourne la solution optimale.*

Preuve. Observons d'abord que, après ⑥, f et π sont des b-flots et potentiels associés à (r, T, L, U).

Prouvons ensuite que la structure d'arbre couvrant est toujours fortement réalisable. Par le choix de δ, la condition $0 \leq f(e) \leq u(e)$ pour tout e reste vérifiée ; donc la structure d'arbre couvrant reste réalisable.

Comme les arcs du sous-chemin de C allant de l'extrémité de e' au pic n'atteignent pas le minimum dans ④, ils garderont une capacité résiduelle positive après l'augmentation.

Pour les arcs du sous-chemin de C allant du pic à l'origine de e', nous devons nous assurer que les arcs inversés de ces arcs ont une capacité résiduelle positive après l'augmentation. C'est évident si $\delta > 0$. Sinon, (si $\delta = 0$) $e \neq e_0$ et, comme la structure d'arbre couvrant était fortement réalisable avant, ni e ni son arc inversé $\overleftarrow{e}$ ne peuvent appartenir à ce sous-chemin (c.-à-d. $e = e_0$ ou $\delta^-(X) \cap E(C) \cap \{e, \overleftarrow{e}\} \neq \emptyset$) et les arcs inversés du sous-chemin de C du pic à l'origine e ou $\overleftarrow{e}$ ont une capacité résiduelle positive.

Par la proposition 9.23, le flot calculé est optimum f quand l'algorithme se termine. Montrons qu'il n'y a pas deux itérations avec le même (f, π), de sorte que chaque structure d'arbre couvrant apparaîtra au plus une fois.

À chaque itération, le coût du flot diminue de $|\gamma|\delta$. Comme $\gamma \neq 0$, considérons les itérations pour lesquelles $\delta = 0$. Dans ce cas, le coût du flot reste constant. Si $e \neq e_0$, alors $e \in L \cap \delta^-(X)$ ou $e \in U \cap \delta^+(X)$, et donc $\sum_{v \in V(G)} \pi(v)$ augmente strictement (par au moins $|\gamma|$). Enfin, si $\delta = 0$ et $e = e_0$, alors $u(e) = 0$, $X = V(G)$, π reste constant, et $|\{e \in L : c_\pi(e) < 0\}| + |\{e \in U : c_\pi(e) > 0\}|$ décroît strictement. Cela montre que les structures d'arbre couvrant sont toutes distinctes durant les itérations. □

Bien que l'ALGORITHME NETWORK SIMPLEX ne soit pas polynomial, il est très efficace en pratique. Orlin [1997] en a proposé une variante polynomiale. Des algorithmes network simplex sur le dual et qui sont polynomiaux ont été proposés par Orlin, Plotkin et Tardos [1993], et Armstrong et Jin [1997].

9.7 Flots dynamiques

Nous allons étudier dans ce dernier paragraphe les flots variant dans le temps (appelés aussi flots dynamiques) ; ici, la valeur du flot pourra varier au cours du temps, et le flot entrant sur un arc arrivera à l'extrémité de cet arc après un délai spécifié :

Définition 9.25. *Soit (G, u, s, t) un réseau avec des temps de transit $l : E(G) \to \mathbb{R}_+$ et un horizon de temps $T \in \mathbb{R}_+$. Un* **flot dynamique de s à t** *f consiste en une fonction Lebesgue-mesurable $f_e : [0, T] \to \mathbb{R}_+$ pour chaque $e \in E(G)$ avec $f_e(\tau) \leq u(e)$ pour tout $\tau \in [0, T]$ et $e \in E(G)$ et*

$$\mathrm{ex}_f(v, a) := \sum_{e \in \delta^-(v)} \int_0^{\max\{0, a - l(e)\}} f_e(\tau) d\tau - \sum_{e \in \delta^+(v)} \int_0^a f_e(\tau) d\tau \geq 0 \quad (9.7)$$

pour tout $v \in V(G) \setminus \{s\}$ et $a \in [0, T]$.

$f_e(\tau)$ est appelé le taux de flot entrant en e à l'instant τ (et quittant cet arc $l(e)$ unités de temps plus tard). L'équation (9.7) permet un stockage intermédiaire du flot sur les sommets, comme pour les préflots de s à t. Il est naturel de chercher à maximiser la quantité de flot arrivant en t :

PROBLÈME DU FLOT DYNAMIQUE MAXIMUM	
Instance	Un réseau (G, u, s, t), des temps de transit $l : E(G) \to \mathbb{R}_+$ et un horizon de temps $T \in \mathbb{R}_+$.
Tâche	Trouver un flot dynamique f de s à t tel que $\mathrm{valeur}(f) := \mathrm{ex}_f(t, T)$ soit maximum.

Suivant Ford et Fulkerson [1958], nous allons montrer comment réduire ce problème au PROBLÈME DU FLOT DE COÛT MINIMUM.

Théorème 9.26. *Toute instance du* PROBLÈME DU FLOT DYNAMIQUE MAXIMUM *peut être transformée en une instance du* PROBLÈME DU FLOT DE COÛT MINIMUM.

Preuve. Soit une instance (G, u, s, t, l, T) du problème du flot dynamique ; définissons un nouvel arc $e' = (t, s)$ et soit $G' := G + e'$. Posons $u(e') := u(E(G))$, $c(e') := -T$ et $c(e) := l(e)$ pour $e \in E(G)$. Considérons l'instance $(G', u, 0, c)$ du PROBLÈME DU FLOT DE COÛT MINIMUM. Soit f' une solution optimale, c.-à-d. une circulation de coût minimum (par rapport à c) dans (G', u). Par la proposition 9.5, f' peut être décomposé en flots sur des circuits ; il existe donc un ensemble de circuits $\mathcal{C}$ dans G' et des nombres positifs $h : \mathcal{C} \to \mathbb{R}_+$ tels que $f'(e) = \sum\{h(C) : C \in \mathcal{C}, e \in E(C)\}$. Nous avons $c(C) \leq 0$ pour tout $C \in \mathcal{C}$ puisque f' est une circulation de coût minimum.

Soit $C \in \mathcal{C}$ avec $c(C) < 0$. C doit contenir e'. Pour $e = (v, w) \in E(C) \setminus \{e'\}$, soit d_e^C la distance de s à v dans (C, c). Soit

$$f_e^*(\tau) := \sum\{h(C) : C \in \mathcal{C}, c(C) < 0, e \in E(C), d_e^C \leq \tau \leq d_e^C - c(C)\}$$

pour $e \in E(G)$ et $\tau \in [0, T]$. Cela définit un flot dynamique de s à t sans stockage intermédiaire (c.-à-d. $\mathrm{ex}_f(v, a) = 0$ pour tout $v \in V(G) \setminus \{s, t\}$ et tout $a \in [0, T]$). De plus,

$$\text{valeur}\,(f^*) \;=\; \sum_{e \in \delta^-(t)} \int_0^{T-l(e)} f_e^*(\tau) d\tau \;=\; -\sum_{e \in E(G')} c(e) f'(e).$$

Montrons que f^* est optimum : soit f un flot dynamique de s à t quelconque, et soit $f_e(\tau) := 0$ pour $e \in E(G)$ et $\tau \notin [0, T]$. Soit $\pi(v) := \mathrm{dist}_{(G'_{f'}, c)}(s, v)$ pour $v \in V(G)$. Comme $G'_{f'}$ ne contient pas de circuit négatif (voir théorème 9.6), π est un potentiel réalisable dans $(G'_{f'}, c)$. Nous avons

$$\text{valeur}\,(f) = \mathrm{ex}_f(t, T) \leq \sum_{v \in V(G)} \mathrm{ex}_f(v, \pi(v))$$

à cause de (9.7), $\pi(t) = T$, $\pi(s) = 0$ et $0 \leq \pi(v) \leq T$ pour tout $v \in V(G)$. Donc

$$\begin{aligned}
\text{valeur}\,(f) &\leq \sum_{e=(v,w) \in E(G)} \left(\int_0^{\pi(w)-l(e)} f_e(\tau) d\tau - \int_0^{\pi(v)} f_e(\tau) d\tau \right) \\
&\leq \sum_{e=(v,w) \in E(G): \pi(w)-l(e) > \pi(v)} (\pi(w) - l(e) - \pi(v)) u(e) \\
&= \sum_{e=(v,w) \in E(G)} (\pi(w) - l(e) - \pi(v)) f'(e) \\
&= \sum_{e=(v,w) \in E(G')} (\pi(w) - c(e) - \pi(v)) f'(e) \\
&= -\sum_{e=(v,w) \in E(G')} c(e) f'(e) \\
&= \text{valeur}\,(f^*).
\end{aligned}$$

□

D'autres problèmes de flot dynamique sont beaucoup plus difficiles. Hoppe et Tardos [2000] ont résolu le problème du transbordement le plus rapide (avec plusieurs sources et puits) avec des temps de transit entiers en utilisant la minimisation de fonctions sous-modulaires (voir chapitre 14). Le problème du flot dynamique de coût minimum est *NP*-difficile (Klinz et Woeginger [2004]). Voir aussi Fleischer et Skutella [2007] pour des algorithmes d'approximation et d'autres informations.

Exercices

1. Montrer que le PROBLÈME DU FLOT MAXIMUM est un cas particulier du PROBLÈME DU FLOT DE COÛT MINIMUM.
2. Soit G un graphe orienté avec des capacités $u : E(G) \to \mathbb{R}_+$, et soit $b : V(G) \to \mathbb{R}$ avec $\sum_{v \in V(G)} b(v) = 0$. Montrer qu'il existe un b-flot si et seulement si

$$\sum_{e \in \delta^+(X)} u(e) \geq \sum_{v \in X} b(v) \quad \text{pour tout } X \subseteq V(G).$$

   (Gale [1957])
3. Soit G un graphe orienté avec des capacités supérieures et inférieures $l, u : E(G) \to \mathbb{R}_+$, tel que $l(e) \leq u(e)$ pour tout $e \in E(G)$, et soit $b_1, b_2 : V(G) \to \mathbb{R}$ avec $b_1(v) \leq b_2(v)$ pour tout $v \in V(G)$. Montrer qu'il existe un flot f vérifiant $l(e) \leq f(e) \leq u(e)$ pour tout $e \in E(G)$ et

$$b_1(v) \leq \sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \leq b_2(v) \quad \text{pour tout } v \in V(G)$$

   si et seulement si

$$\sum_{e \in \delta^+(X)} u(e) \geq \max \left\{ \sum_{v \in X} b_1(v), - \sum_{v \in V(G) \setminus X} b_2(v) \right\} + \sum_{e \in \delta^-(X)} l(e)$$

   pour tout $X \subseteq V(G)$. (Cela est une généralisation de l'exercice 4 du chapitre 8 et de l'exercice 2 de ce chapitre.)
   (Hoffman [1960])
4. Montrer le théorème suivant dû à Ore [1956]. Soit un graphe orienté G et des entiers non négatifs $a(x), b(x)$ pour chaque $x \in V(G)$; G a un sous-graphe couvrant H avec $|\delta_H^+(x)| = a(x)$ et $|\delta_H^-(x)| = b(x)$ pour tout $x \in V(G)$ si et seulement si

$$\sum_{x \in V(G)} a(x) = \sum_{x \in V(G)} b(x) \quad \text{et}$$

$$\sum_{x\in X} a(x) \;\leq\; \sum_{y\in V(G)} \min\{b(y), |E_G^+(X, \setminus\{y\})|\} \quad \text{pour tout } X \subseteq V(G).$$

(Ford et Fulkerson [1962])

5. Considérons le PROBLÈME DU FLOT DE COÛT MINIMUM avec des capacités infinies ($u(e) = \infty$) autorisées sur certains arcs e.
   (a) Montrer qu'une instance est non bornée si et seulement si elle est réalisable et s'il existe un circuit négatif dont tous les arcs ont une capacité infinie.
   (b) Montrer comment vérifier en un temps $O(n^3 + m)$ si une instance est non bornée.
   (c) Montrer que, dans toute instance qui n'est pas non bornée, chaque capacité infinie peut être remplacée par une capacité finie.

* 6. Soit (G, u, c, b) une instance du PROBLÈME DU FLOT DE COÛT MINIMUM. Une fonction $\pi : V(G) \to \mathbb{R}$ sera appelée potentiel optimal s'il existe un b-flot f de coût minimum tel que π soit un potentiel réalisable par rapport à (G_f, c).
   (a) Montrer qu'une fonction $\pi : V(G) \to \mathbb{R}$ est un potentiel optimal si et seulement si pour tout $X \subseteq V(G)$:

$$b(X) + \sum_{e\in\delta^-(X):c_\pi(e)<0} u(e) \;\leq\; \sum_{e\in\delta^+(X):c_\pi(e)\leq 0} u(e).$$

   (b) Soit $\pi : V(G) \to \mathbb{R}$ donné ; montrer comment trouver un ensemble X violant la condition (a) ou prouver qu'aucun ensemble X ne viole cette condition.
   (c) Supposons qu'un potentiel optimal soit donné ; montrer comment trouver un b-flot en un temps $O(n^3)$.

   Note : cela conduit à l'algorithme connu sous le nom d'algorithme d'annulation des coupes pour le PROBLÈME DU FLOT DE COÛT MINIMUM.
   (Hassin [1983])

7. Considérons le schéma d'algorithme suivant pour le PROBLÈME DU FLOT DE COÛT MINIMUM : partir d'un b-flot, puis, aussi longtemps qu'il existe un circuit augmentant négatif, augmenter le flot le long de ce circuit (par la plus grande quantité possible). Nous avons vu au paragraphe 9.3 que nous obtenons un algorithme fortement polynomial si nous choisissons toujours un circuit de poids moyen minimum. Montrer que sans cette condition, on ne peut pas garantir que l'algorithme se termine.
   (Utiliser la construction de l'exercice 2 du chapitre 8.)

8. Considérons le problème de l'exercice 3 avec, en plus, une fonction poids $c : E(G) \to \mathbb{R}$. Peut-on trouver un flot de coût minimum qui satisfait les contraintes de l'exercice 3 ? (Ramener ce problème à un PROBLÈME DU FLOT DE COÛT MINIMUM standard.)

9. Le PROBLÈME DU POSTIER CHINOIS ORIENTÉ peut être ainsi formulé : soit un graphe simple orienté G avec des poids $c : E(G) \to \mathbb{R}_+$, trouver $f : E(G) \to \mathbb{N}$ tel que le graphe qui contient $f(e)$ copies de chaque arc $e \in E(G)$ soit eulérien et tel que $\sum_{e \in E(G)} c(e)f(e)$ soit minimum. Comment résoudre ce problème en temps polynomial ?
(Pour le PROBLÈME DU POSTIER CHINOIS NON ORIENTÉ, voir le paragraphe 12.2.)

* 10. Le problème du b-couplage fractionnaire est défini de la manière suivante : étant donné un graphe non orienté G, des capacités $u : E(G) \to \mathbb{R}_+$, des nombres $b : V(G) \to \mathbb{R}_+$ et des poids $c : E(G) \to \mathbb{R}$, nous cherchons $f : E(G) \to \mathbb{R}_+$ avec $f(e) \leq u(e)$ pour tout $e \in E(G)$ et $\sum_{e \in \delta(v)} f(e) \leq b(v)$ pour tout $v \in V(G)$ tel que $\sum_{e \in E(G)} c(e)f(e)$ soit maximum.

   (a) Montrer comment résoudre ce problème en le réduisant au PROBLÈME DU FLOT DE COÛT MINIMUM.

   (b) Supposons que b et u soient entiers. Montrer que le problème du b-couplage fractionnaire a toujours une solution demi-entière f (c.-à-d. $2f(e) \in \mathbb{Z}$ pour tout $e \in E(G)$).

   Note : le PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM (entier) fera l'objet du paragraphe 12.1.

* 11. Trouver un algorithme combinatoire polynomial pour le problème de l'empilement d'intervalles défini à l'exercice 16 du chapitre 5.
(Arkin et Silverberg [1987])

12. Soit un programme linéaire $\max\{cx : Ax \leq b\}$ où tous les coefficients de A valent -1, 0, ou $+1$, et où chaque colonne de A contient au plus un seul $+1$ et au plus un seul -1. Montrer que ce PL est équivalent à une instance du PROBLÈME DU FLOT DE COÛT MINIMUM.

13. Montrer que l'ALGORITHME PAR PLUS COURTS CHEMINS SUCCESSIFS répond correctement à la question de l'existence d'un b-flot.

14. La technique de réduction d'échelle introduite aux paragraphes 8.4 et 9.4 s'applique dans un cadre très général : soit Ψ une famille de systèmes d'ensembles, chacun de ces systèmes contenant l'ensemble vide. Supposons qu'il existe un algorithme qui résout le problème suivant : soit $(E, \mathcal{F}) \in \Psi$, des poids $c : E \to \mathbb{Z}_+$ et un ensemble $X \in \mathcal{F}$; trouver $Y \in \mathcal{F}$ tel que $c(Y) > c(X)$ ou certifier qu'un tel ensemble Y n'existe pas. Supposons que cet algorithme ait un temps de calcul polynomial par rapport à $\text{taille}(c)$. Montrer qu'il existe un algorithme qui trouve un ensemble de poids maximum $X \in \mathcal{F}$ dans $(E, \mathcal{F}) \in \Psi$ et $c : E \to \mathbb{Z}_+$, en un temps polynomial par rapport à $\text{taille}(c)$.
(Grötschel et Lovász [1995] ; voir aussi Schulz, Weismantel et Ziegler [1995], et Schulz et Weismantel [2002])

15. Montrer que l'ALGORITHME D'ORLIN trouve toujours une solution qui est un arbre couvrant.

16. Montrer que dans ⑦ de l'ALGORITHME D'ORLIN on peut remplacer la borne $8n\gamma$ par $5n\gamma$.

17. Considérons les calculs de plus courts chemins avec des poids non négatifs (en utilisant l'ALGORITHME DE DIJKSTRA) dans les algorithmes des paragraphes 9.4 et 9.5. Montrer que même dans le cas d'arcs parallèles, chacun de ces calculs peut se faire en $O(n^2)$, pourvu que la liste d'incidence de G soit triée par rapport aux coûts des arcs. Conclure que l'ALGORITHME D'ORLIN a un temps de calcul $O(mn^2 \log m)$.

* 18. L'ALGORITHME PUSH-RELABEL (paragraphe 8.5) peut se généraliser au PROBLÈME DU FLOT DE COÛT MINIMUM. Soit (G, u, b, c) une instance avec des coûts entiers c ; nous cherchons un b-flot f et un potentiel réalisable π de (G_f, c). Commençons par poser $\pi := 0$ et saturons tous les arcs e de coût négatif. Puis appliquons ③ de l'ALGORITHME PUSH-RELABEL avec les modifications suivantes : un arc e est admissible si $e \in E(G_f)$ et $c_\pi(e) < 0$. Un sommet v est actif si $b(v) + \mathrm{ex}_f(v) > 0$. L'opération RELABEL (v) consiste à poser $\pi(v) := \max\{\pi(w) - c(e) - 1 : e = (v, w) \in E(G_f)\}$. L'opération PUSH (e) pour $e \in \delta^+(v)$ consiste à poser $\gamma := \min\{b(v) + \mathrm{ex}_f(v), u_f(e)\}$.

   (a) Montrer que le nombre d'opérations RELABEL est $O(n^2|c_{\max}|)$, où $c_{\max} = \max_{e \in E(G)} c(e)$.
   Indication : un sommet w avec $b(w) + \mathrm{ex}_f(w) < 0$ doit être connecté dans G_f à tout sommet actif v. Noter que $b(w)$ ne change jamais et voir les preuves des lemmes 8.22 et 8.24.

   (b) Montrer que le temps total de calcul est $O(n^2 m c_{\max})$.

   (c) Montrer que l'algorithme calcule une solution optimale.

   (d) Appliquer la technique de réduction d'échelle pour obtenir un algorithme en $O(n^2 m \log c_{\max})$ pour le PROBLÈME DU FLOT DE COÛT MINIMUM avec des coûts entiers c.

   (Goldberg et Tarjan [1990])

19. Soit (G, u, c, b) une instance du PROBLÈME DU FLOT DE COÛT MINIMUM. Soit $\bar{e} \in E(G)$ avec $c(\bar{e}) > (|V(G)| - 1) \max_{e \in E(G) \setminus \{\bar{e}\}} |c(e)|$. Montrer le résultat suivant : s'il existe un b-flot f dans (G, u) avec $f(\bar{e}) = 0$, alors $f(\bar{e}) = 0$ est vérifié pour toute solution optimale f.

20. Soit un réseau (G, u, s, t) avec des temps de transit $l : E(G) \to \mathbb{Z}_+$, un horizon de temps $T \in \mathbb{N}$, une valeur $V \in \mathbb{R}_+$, et des coûts $c : E(G) \to \mathbb{R}_+$. Nous cherchons un flot dynamique f de s à t avec $\mathrm{valeur}\,(f) = V$ et de coût minimum $\sum_{e \in E(G)} c(e) \int_0^T f_e(\tau) d\tau$. Montrer comment résoudre ce problème en temps polynomial quand T est constant.
   Indication : considérer un réseau étendu dans le temps avec une copie de G à chaque pas discret de temps.

Références

Littérature générale :

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. [1993] : Network Flows. Prentice-Hall, Englewood Cliffs 1993

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. [1998] : Combinatorial Optimization. Wiley, New York 1998, Chapter 4

Goldberg, A.V., Tardos, É., Tarjan, R.E. [1990] : Network flow algorithms. In : Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, eds.), Springer, Berlin 1990, pp. 101–164

Gondran, M., Minoux, M. [1984] : Graphs and Algorithms. Wiley, Chichester 1984, Chapter 5

Jungnickel, D. [2007] : Graphs, Networks and Algorithms. Third Edition. Springer, Berlin 2007, Chapters 10 and 11

Lawler, E.L. [1976] : Combinatorial Optimization : Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Chapter 4

Ruhe, G. [1991] : Algorithmic Aspects of Flows in Networks. Kluwer Academic Publishers, Dordrecht 1991

Références citées :

Arkin, E.M., Silverberg, E.B. [1987] : Scheduling jobs with fixed start and end times. Discrete Applied Mathematics 18 (1987), 1–8

Armstrong, R.D., Jin, Z. [1997] : A new strongly polynomial dual network simplex algorithm. Mathematical Programming 78 (1997), 131–148

Busacker, R.G., Gowen, P.J. [1961] : A procedure for determining a family of minimum-cost network flow patterns. ORO Technical Paper 15, Operational Research Office, Johns Hopkins University, Baltimore 1961

Cunningham, W.H. [1976] : A network simplex method. Mathematical Programming 11 (1976), 105–116

Dantzig, G.B. [1951] : Application of the simplex method to a transportation problem. In : Activity Analysis and Production and Allocation (T.C. Koopmans, Ed.), Wiley, New York 1951, pp. 359–373

Edmonds, J., Karp, R.M. [1972] : Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM 19 (1972), 248–264

Fleischer, L., Skutella, M. [2007] : Quickest flows over time. SIAM Journal on Computing 36 (2007), 1600–1630

Ford, L.R., Fulkerson, D.R. [1958] : Constructing maximal dynamic flows from static flows. Operations Research 6 (1958), 419–433

Ford, L.R., Fulkerson, D.R. [1962] : Flows in Networks. Princeton University Press, Princeton 1962

Gale, D. [1957] : A theorem on flows in networks. Pacific Journal of Mathematics 7 (1957), 1073–1082

Goldberg, A.V., Tarjan, R.E. [1989] : Finding minimum-cost circulations by cancelling negative cycles. Journal of the ACM 36 (1989), 873–886

Goldberg, A.V., Tarjan, R.E. [1990] : Finding minimum-cost circulations by successive approximation. Mathematics of Operations Research 15 (1990), 430–466

Grötschel, M., Lovász, L. [1995] : Combinatorial optimization. In : Handbook of Combinatorics ; Vol. 2 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam 1995

Hassin, R. [1983] : The minimum cost flow problem : a unifying approach to dual algorithms and a new tree-search algorithm. Mathematical Programming 25 (1983), 228–239

Hitchcock, F.L. [1941] : The distribution of a product from several sources to numerous localities. Journal of Mathematical Physics 20 (1941), 224–230

Hoffman, A.J. [1960] : Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. In : Combinatorial Analysis (R.E. Bellman, M. Hall, eds.), AMS, Providence 1960, pp. 113–128

Hoppe, B., Tardos, É. [2000] : The quickest transshipment problem. Mathematics of Operations Research 25 (2000), 36–62

Iri, M. [1960] : A new method for solving transportation-network problems. Journal of the Operations Research Society of Japan 3 (1960), 27–87

Jewell, W.S. [1958] : Optimal flow through networks. Interim Technical Report 8, MIT 1958

Klein, M. [1967] : A primal method for minimum cost flows, with applications to the assignment and transportation problems. Management Science 14 (1967), 205–220

Klinz, B., Woeginger, G.J. [2004] : Minimum cost dynamic flows : the series-parallel case. Networks 43 (2004), 153–162

Orden, A. [1956] : The transshipment problem. Management Science 2 (1956), 276–285

Ore, O. [1956] : Studies on directed graphs I. Annals of Mathematics 63 (1956), 383–406

Orlin, J.B. [1993] : A faster strongly polynomial minimum cost flow algorithm. Operations Research 41 (1993), 338–350

Orlin, J.B. [1997] : A polynomial time primal network simplex algorithm for minimum cost flows. Mathematical Programming 78 (1997), 109–129

Orlin, J.B., Plotkin, S.A., Tardos, É. [1993] : Polynomial dual network simplex algorithms. Mathematical Programming 60 (1993), 255–276

Plotkin, S.A., Tardos, É. [1990] : Improved dual network simplex. Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (1990), 367–376

Schulz, A.S., Weismantel, R., Ziegler, G.M. [1995] : 0/1-Integer Programming : optimization and augmentation are equivalent. In : Algorithms – ESA '95 ; LNCS 979 (P. Spirakis, ed.), Springer, Berlin 1995, pp. 473–483

Schulz, A.S., Weismantel, R. [2002] : The complexity of generic primal algorithms for solving general integer problems. Mathematics of Operations Research 27 (2002), 681–192

Tardos, É. [1985] : A strongly polynomial minimum cost circulation algorithm. Combinatorica 5 (1985), 247–255

Tolstoĭ, A.N. [1930] : Metody nakhozhdeniya naimen'shego summovogo kilometrazha pri planirovanii perevozok v prostanstve. In : Planirovanie Perevozok, Sbornik pervyĭ, Transpechat' NKPS, Moskow 1930, pp. 23–55. (See A. Schrijver, On the history of the transportation and maximum flow problems, Mathematical Programming 91 (2002), 437–445)

Tomizawa, N. [1971] : On some techniques useful for solution of transportation network problems. Networks 1 (1971), 173–194

Vygen, J. [2002] : On dual minimum cost flow algorithms. Mathematical Methods of Operations Research 56 (2002), 101–126

Wagner, H.M. [1959] : On a class of capacitated transportation problems. Management Science 5 (1959), 304–318

Chapitre 10

Couplage maximum

La théorie du couplage est un sujet classique et très important de la théorie des graphes et de l'optimisation combinatoire. Tous les graphes dans ce chapitre seront non orientés. Rappelons qu'un couplage est un ensemble d'arêtes deux à deux non incidentes à un même sommet. Notre problème est le suivant :

PROBLÈME DU COUPLAGE MAXIMUM	
Instance	Un graphe non orienté G.
Tâche	Trouver un couplage de cardinalité maximum dans G.

La version pondérée de ce problème, plus difficile, sera étudiée au chapitre 11. La version non pondérée a aussi des applications : supposons que, dans le PROBLÈME D'AFFECTATION DES TÂCHES, chaque tâche ait le même temps d'exécution, par exemple une heure, et que nous cherchions à terminer toutes les tâches en une heure. Plus précisément, étant donné un graphe biparti G ayant pour bipartition $V(G) = A \cup B$, existe-t-il $x : E(G) \to \mathbb{R}_+$ tel que $\sum_{e \in \delta(a)} x(e) = 1$ pour toute tâche $a \in A$ et $\sum_{e \in \delta(b)} x(e) \leq 1$ pour chaque employé $b \in B$. Cela peut s'écrire comme un système d'inégalités linéaires $x \geq 0$, $Mx \leq \mathbb{1}$, $M'x \geq \mathbb{1}$, où les lignes de M et de M' sont les lignes de la matrice d'incidence sommet-arête de G. Ces matrices sont totalement unimodulaires (théorème 5.25). Par le théorème 5.20, nous en déduisons que, s'il existe une solution x, il existe aussi une solution entière. Observons que les solutions entières du système d'inégalités précédent sont les vecteurs d'incidence des couplages de G couvrant A.

Définition 10.1. *Soit G un graphe et soit M un couplage de G. Nous dirons qu'un sommet v est* **couvert** *par M s'il existe une arête $e \in M$ incidente à v. Le couplage M est un* **couplage parfait** *si tous les sommets sont couverts par M.*

Au paragraphe 10.1 nous étudierons les couplages dans les graphes bipartis. Algorithmiquement, ce problème se ramène au PROBLÈME DU FLOT MAXIMUM. Le théorème flot-max/coupe-min ainsi que la notion de chemin augmentant ont dans ce contexte des interprétations élégantes.

Le couplage, en général, ne se réduit pas à un problème de flot. Nous présenterons aux paragraphes 10.2 et 10.3 deux conditions nécessaires et suffisantes pour qu'un graphe ait un couplage parfait. Au paragraphe 10.4 nous étudierons les graphes facteur-critiques couvrant tous les sommets à l'exception d'un seul v, quel que soit $v \in V(G)$. Ces graphes jouent un rôle important dans l'algorithme d'Edmonds du COUPLAGE DE CARDINALITÉ MAXIMUM, présenté au paragraphe 10.5, ainsi que dans sa version avec poids présentée aux paragraphes 11.2 et 11.3.

10.1 Couplage dans les graphes bipartis

Puisque le PROBLÈME DU COUPLAGE DE CARDINALITÉ MAXIMUM est plus simple quand G est biparti, nous étudierons d'abord ce cas. Dans ce paragraphe, la bipartition $V(G) = A \dot\cup B$ de G sera supposée connue. Puisque nous pouvons supposer G connexe, cette bipartition est unique (exercice 20 du chapitre 2).

Si G est un graphe, soit $\nu(G)$ la cardinalité maximum d'un couplage de G, et soit $\tau(G)$ la cardinalité minimum d'une couverture par les sommets de G.

Théorème 10.2. (König [1931]) *Si G est biparti, alors $\nu(G) = \tau(G)$.*

Preuve. Soit le graphe $G' = (V(G) \dot\cup \{s,t\},\ E(G) \cup \{(s,a) : a \in A\} \cup \{(b,t) : b \in B\})$. $\nu(G)$ est le nombre maximum de chaînes de s à t sommet-disjointes, tandis que $\tau(G)$ est le nombre minimum de sommets dont la suppression déconnecte s et t. Le théorème se déduit alors du théorème de Menger 8.10. □

La relation $\nu(G) \leq \tau(G)$ est vraie pour tout graphe (biparti ou non), mais nous n'avons pas en général l'égalité (comme le triangle K_3 le montre).

De nombreux énoncés sont équivalents au théorème de König. Le théorème de Hall est certainement la variante la plus connue.

Théorème 10.3. (Hall [1935]) *Soit G un graphe biparti avec bipartition $V(G) = A \dot\cup B$. Alors G a un couplage qui couvre A si et seulement si*

$$|\Gamma(X)| \ \geq\ |X| \qquad \textit{pour tout } X \subseteq A. \tag{10.1}$$

Preuve. La condition est manifestement nécessaire. Pour montrer qu'elle est suffisante, supposons qu'il n'existe pas de couplage de G couvrant A, c.-à-d. tel que $\nu(G) < |A|$. Par le théorème 10.2 cela implique $\tau(G) < |A|$.

Soit $A' \subseteq A,\ B' \subseteq B$ tel que $A' \cup B'$ couvre toutes les arêtes et $|A' \cup B'| < |A|$. Alors il est évident que $\Gamma(A \setminus A') \subseteq B'$. Donc $|\Gamma(A \setminus A')| \leq |B'| < |A| - |A'| = |A \setminus A'|$, et la condition de Hall (10.1) est violée. □

Il n'est pas difficile de démontrer le théorème de Hall directement. Voici une autre preuve donnée par Halmos et Vaughan [1950] :

Deuxième démonstration du théorème 10.3 : montrons que, quand la condition (10.1) est vérifiée, G a un couplage couvrant A (induction sur $|A|$).

Les cas $|A| = 0$ et $|A| = 1$ sont triviaux ; si $|A| \geq 2$, nous étudierons deux cas : si $|\Gamma(X)| > |X|$ pour tout ensemble propre non vide X de A, prenons une arête quelconque (a, b) ($a \in A$, $b \in B$), supprimons ces deux sommets et utilisons l'induction. Le nouveau graphe, plus petit, satisfait la condition de Hall puisque $|\Gamma(X)| - |X|$ a diminué d'au plus un pour tout $X \subseteq A \setminus \{a\}$.

Supposons maintenant qu'il y ait un sous-ensemble propre non vide X de A avec $|\Gamma(X)| = |X|$. Par induction, il existe un couplage couvrant X dans $G[X \cup \Gamma(X)]$. Montrons que ce couplage s'étend à un couplage de G couvrant A. Grâce à l'induction, montrons que $G[(A \setminus X) \cup (B \setminus \Gamma(X))]$ vérifie la condition de Hall. En effet, pour tout $Y \subseteq A \setminus X$ nous avons (dans le graphe original G) :

$$|\Gamma(Y) \setminus \Gamma(X)| \;=\; |\Gamma(X \cup Y)| - |\Gamma(X)| \;\geq\; |X \cup Y| - |X| \;=\; |Y|.$$

□

Un cas particulier du théorème de Hall est le «théorème du mariage» :

Théorème 10.4. (Frobenius [1917]) *Soit G un graphe biparti avec bipartition $V(G) = A \,\dot\cup\, B$. Alors G a un couplage parfait si et seulement si $|A| = |B|$ et $|\Gamma(X)| \geq |X|$ pour tout $X \subseteq A$.* □

Des applications du théorème de Hall sont proposées dans les exercices 4-7.

La preuve du théorème de König 10.2 montre comment résoudre algorithmiquement le problème du couplage biparti :

Théorème 10.5. *Le* PROBLÈME DU COUPLAGE MAXIMUM *pour les graphes bipartis G se résout en un temps $O(nm)$ ($n = |V(G)|$, $m = |E(G)|$).*

Preuve. Soit G un graphe biparti avec bipartition $V(G) = A \,\dot\cup\, B$. Ajoutons un sommet s connecté à tous les sommets de A, et ajoutons un autre sommet t connecté à tous les sommets de B. Orientons les arêtes de s vers A, de A à B, et de B à t. Posons toutes les capacités égales à 1. Alors un flot maximum de s à t est associé à un couplage de cardinalité maximum (et inversement).

Nous pouvons donc utiliser l'ALGORITHME DE FORD-FULKERSON et trouver un flot maximum de s à t (et un couplage maximum) après au plus n augmentations. Chaque augmentation prend un temps $O(m)$, ce qui démontre le théorème. □

Ce résultat est essentiellement dû à Kuhn [1955]. De ce fait, nous pouvons encore utiliser le concept de plus court chemin augmentant (voir l'ALGORITHME D'EDMONDS-KARP). On obtient ainsi l'algorithme en $O\left(\sqrt{n}(m+n)\right)$ de Hopcroft et Karp [1973]. Cet algorithme sera étudié dans les exercices 9 et 10. De légères améliorations de l'ALGORITHME DE HOPCROFT-KARP conduisent à des temps de calcul $O\left(n\sqrt{\frac{mn}{\log n}}\right)$ (Alt *et al.* [1991]) et $O\left(m\sqrt{n}\frac{\log(n^2/m)}{\log n}\right)$ (Feder et Motwani [1995]). La dernière borne est la meilleure connue pour les graphes denses.

Reformulons la notion de chemin augmentant dans notre contexte.

Définition 10.6. *Soit G un graphe (biparti ou non) ; soit M un couplage de G. Une chaîne P est une* **chaîne *M*-alternée** *si $E(P) \setminus M$ est un couplage. Une chaîne M-alternée est* ***M*-augmentante** *si ses extrémités ne sont pas couvertes par M.*

Une chaîne M-augmentante a manifestement une longueur impaire.

Théorème 10.7. (Berge [1957]) *Soit G un graphe (biparti ou non) ; soit M un couplage de G : M est maximum si et seulement s'il n'existe pas de chaîne M-augmentante.*

Preuve. S'il existe une chaîne M-augmentante P, $M \triangle E(P)$ est un couplage de cardinalité supérieure à celle de M et M n'est pas maximum. D'autre part, s'il existe un couplage M' tel que $|M'| > |M|$, $M \triangle M'$ est l'union de chaînes et cycles sommet-disjoints, et une de ces chaînes est M-augmentante. □

Si G est biparti, le théorème de Berge se déduit aussi du théorème 8.5.

10.2 Matrice de Tutte

Considérons maintenant les couplages d'un point de vue algébrique. Soit G un graphe simple non orienté, et soit G' le graphe orienté résultant d'une orientation arbitraire des arêtes de G. Pour tout vecteur $x = (x_e)_{e \in E(G)}$ de variables, définissons la **matrice de Tutte**

$$T_G(x) \;=\; (t^x_{vw})_{v,w \in V(G)}$$

de la manière suivante :

$$t^x_{vw} \;:=\; \begin{cases} x_{(v,w)} & \text{si } (v,w) \in E(G') \\ -x_{(v,w)} & \text{si } (w,v) \in E(G') \\ 0 & \text{sinon} \end{cases} .$$

(Une telle matrice M, où $M = -M^\top$, est appelée **antisymétrique**.) $\det T_G(x)$ est un polynôme par rapport aux variables x_e ($e \in E(G)$).

Théorème 10.8. (Tutte [1947]) *G possède un couplage parfait si et seulement si $\det T_G(x)$ n'est pas identiquement égal à zéro.*

Preuve. Soit $V(G) = \{v_1, \ldots, v_n\}$, et soit S_n l'ensemble des permutations sur $\{1, \ldots, n\}$. Par la définition d'un déterminant,

$$\det T_G(x) \;=\; \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} t^x_{v_i, v_{\pi(i)}} .$$

Soit $S'_n := \left\{ \pi \in S_n : \prod_{i=1}^n t^x_{v_i, v_{\pi(i)}} \neq 0 \right\}$. Chaque permutation $\pi \in S_n$ correspond à un graphe orienté $H_\pi := (V(G), \{(v_i, v_{\pi(i)}) : i = 1, \ldots, n\})$ où

$|\delta^-_{H_\pi}(x)| = |\delta^+_{H_\pi}(x)| = 1$ pour $x \in V(G)$. Si $\pi \in S'_n$, H_π est un sous-graphe de $\overleftrightarrow{G'}$.

S'il existe une permutation $\pi \in S'_n$ telle que H_π soit une union de cycles pairs, on obtiendra un couplage parfait de G en prenant les arcs d'indice pair dans la description de chacun des cycles et en ignorant les orientations des arcs.

Sinon il existe, pour chaque $\pi \in S'_n$, une permutation $r(\pi) \in S'_n$ telle que $H_{r(\pi)}$ est obtenue en inversant le premier cycle impair dans H_π, c.-à-d. le cycle impair contenant le sommet de plus petit indice. Notons que $r(r(\pi)) = \pi$.

De plus, $\mathrm{sgn}(\pi) = \mathrm{sgn}(r(\pi))$ (les deux permutations ont la même signature) : si les sommets du premier cycle impair sont $w_1, \ldots, w_{2k+1}$ avec $\pi(w_i) = w_{i+1}$ $(i = 1, \ldots, 2k)$ et $\pi(w_{2k+1}) = w_1$, alors $r(\pi)$ s'obtient par $2k$ transpositions : pour $j = 1, \ldots, k$ échanger $\pi(w_{2j-1})$ avec $\pi(w_{2k})$ et ensuite $\pi(w_{2j})$ avec $\pi(w_{2k+1})$.

Comme $\prod_{i=1}^n t^x_{v_i, v_{\pi(i)}} = -\prod_{i=1}^n t^x_{v_i, v_{r(\pi)(i)}}$, les deux termes correspondants dans la somme

$$\det T_G(x) = \sum_{\pi \in S'_n} \mathrm{sgn}(\pi) \prod_{i=1}^n t^x_{v_i, v_{\pi(i)}}$$

s'annulent l'un l'autre. Comme cela est vrai pour toutes les paires $\pi, r(\pi) \in S'_n$, $\det T_G(x)$ est identiquement nul.

Donc, $\det T_G(x)$ est identiquement nul quand G n'a pas de couplage parfait. Inversement, soit M un couplage parfait de G et soit π la permutation définie par : $\pi(i) := j$ et $\pi(j) := i$ pour tout $(v_i, v_j) \in M$. Le terme correspondant $\prod_{i=1}^n t^x_{v_i, v_{\pi(i)}} = \prod_{e \in M} \left(-{x_e}^2\right)$ ne peut s'annuler avec un autre terme et $\det T_G(x)$ n'est pas uniquement nul. □

Tutte a utilisé le théorème 10.8 pour montrer son principal théorème sur les couplages, le théorème 10.13. Le théorème 10.8 ne donne pas une bonne caractérisation de l'existence d'un couplage parfait dans un graphe, car un déterminant est facile à calculer quand les coefficients sont des nombres (théorème 4.10), mais difficile à calculer quand ce sont des variables. Mais le théorème suggère un algorithme randomisé pour le PROBLÈME DU COUPLAGE MAXIMUM :

Corollaire 10.9. (Lovász [1979]) *Soit $x = (x_e)_{e \in E(G)}$ un vecteur aléatoire où chaque coordonnée suit la loi uniforme sur l'intervalle $[0, 1]$. Alors, avec probabilité 1, le rang de $T_G(x)$ est égal à deux fois la taille d'un couplage maximum.*

Preuve. Supposons que le rang de $T_G(x)$ soit k ; on peut supposer que les k premières lignes sont linéairement indépendantes. Puisque $T_G(x)$ est antisymétrique, les k premières colonnes sont aussi linéairement indépendantes. La sous-matrice principale $(t^x_{v_i, v_j})_{1 \le i,j \le k}$ est non singulière et le sous-graphe $G[\{v_1, \ldots, v_k\}]$ a un couplage parfait par le théorème 10.8. En particulier, k est pair et G a un couplage de cardinalité $\frac{k}{2}$.

Inversement, si G a un couplage de cardinalité k, le déterminant de la sous-matrice principale T' dont les lignes et colonnes correspondent aux $2k$ sommets couverts par M n'est pas identiquement nul par le théorème 10.8. L'ensemble des

vecteurs x tels que $\det T'(x) = 0$ est de mesure nulle. Donc, avec probabilité 1, le rang de $T_G(x)$ est au moins $2k$. □

Il n'est pas possible de choisir des nombres aléatoires dans $[0, 1]$ avec un ordinateur digital. Cependant, on peut montrer qu'il suffit de choisir des entiers aléatoires dans l'ensemble fini $\{1, 2, \ldots, N\}$. Pour N suffisamment grand, la probabilité d'erreur deviendra arbitrairement petite (voir Lovász [1979]). L'algorithme de Lovász peut être également utilisé pour trouver un couplage maximum (pas seulement sa cardinalité). Voir Rabin et Vazirani [1989], Mulmuley, Vazirani et Vazirani [1987], et Mucha et Sankowski [2004] pour d'autres algorithmes randomisés pour trouver un couplage maximum dans un graphe. Notons également que Geelen [2000] a montré comment dérandomiser l'algorithme de Lovász. Bien que la complexité soit moins bonne que celle de l'algorithme d'Edmonds (voir paragraphe 10.5), cette approche est importante pour certaines généralisations du PROBLÈME DU COUPLAGE MAXIMUM (voir par exemple Geelen et Iwata [2005]).

10.3 Théorème de Tutte

Nous considérons maintenant le PROBLÈME DU COUPLAGE MAXIMUM dans tous les graphes. Une condition nécessaire pour qu'un graphe ait un couplage parfait est que chaque composante connexe soit paire (c.-à-d. ait un nombre pair de sommets). Cette condition n'est pas suffisante, comme le montre le graphe $K_{1,3}$ (figure 10.1(a)).

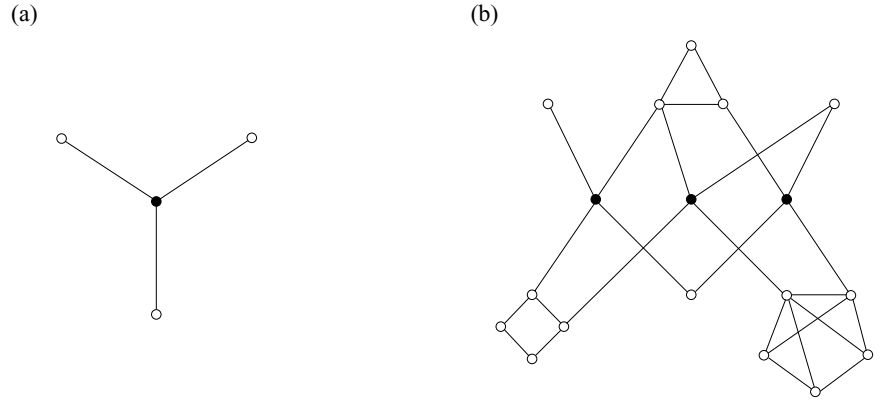


Figure 10.1.

On peut voir que $K_{1,3}$ n'a pas de couplage parfait parce qu'il y a un sommet (noir sur la figure) dont la suppression produit trois composantes connexes impaires. Le graphe de la figure 10.1(b) est plus compliqué. Ce graphe contient-il un couplage

parfait ? Si nous supprimons les trois sommets noirs, nous obtenons cinq composantes connexes impaires (et une paire). S'il existait un couplage parfait, au moins un sommet de chaque composante connexe impaire devrait être couplé avec un des sommets noirs. C'est impossible, car le nombre de composantes connexes impaires excède le nombre de sommets noirs.

Plus généralement, soit $q_G(X)$ le nombre de composantes connexes de $G - X$ pour $X \subseteq V(G)$. Un graphe tel que $q_G(X) > |X|$ pour un sous-ensemble $X \subseteq V(G)$ ne peut pas avoir de couplage parfait : sinon il faudrait que, pour chaque composante connexe impaire de $G - X$, il y ait au moins une arête du couplage qui connecterait cette composante connexe avec X, ce qui est impossible s'il y a plus de composantes connexes impaires que d'éléments de X. Le théorème de Tutte affirme que cette condition nécessaire est aussi suffisante :

Définition 10.10. *Un graphe G satisfait la* **condition de Tutte** *si $q_G(X) \leq |X|$ pour $X \subseteq V(G)$. Un sous-ensemble $X \subseteq V(G)$ est une* **barrière** *si $q_G(X) = |X|$.*

Pour montrer que la condition de Tutte est suffisante, nous aurons besoin d'une observation simple et d'une importante définition :

Proposition 10.11. *Pour tout graphe G et tout $X \subseteq V(G)$,*

$$q_G(X) - |X| \equiv |V(G)| \pmod 2.$$

□

Définition 10.12. *Un graphe G sera appelé* **facteur-critique** *si $G - v$ a un couplage parfait pour tout $v \in V(G)$. Un couplage sera dit* **presque parfait** *s'il couvre tous les sommets sauf un.*

Nous pouvons maintenant démontrer le théorème de Tutte :

Théorème 10.13. (Tutte [1947]) *Un graphe G a un couplage parfait si et seulement s'il satisfait la condition de Tutte :*

$$q_G(X) \leq |X| \qquad \text{pour tout } X \subseteq V(G).$$

Preuve. Nous avons déjà montré que la condition est nécessaire. Montrons qu'elle est suffisante par induction sur $|V(G)|$ (le cas $|V(G)| \leq 2$ étant trivial).

Soit G un graphe vérifiant la condition de Tutte. $|V(G)|$ ne peut être impair sinon la condition de Tutte serait violée puisque $q_G(\emptyset) \geq 1$.

Donc par la proposition 10.11, $|X| - q_G(X)$ est pair pour tout $X \subseteq V(G)$. Puisque $|V(G)|$ est pair et puisque la condition de Tutte est vérifiée, chaque singleton est une barrière.

Choisissons une barrière maximale X. $G - X$ a $|X|$ composantes connexes impaires. $G - X$ ne peut avoir de composante connexe paire sinon $X \cup \{v\}$, où v est un sommet d'une composante connexe paire, est aussi une barrière ($G - (X \cup \{v\})$ a $|X| + 1$ composantes connexes impaires), contredisant la maximalité de X.

Montrons maintenant que chaque composante connexe impaire de $G - X$ est facteur-critique. En effet, soit C une composante connexe de $G - X$ et soit $v \in V(C)$. Si $C - v$ n'a pas de couplage parfait, il existe par l'hypothèse d'induction $Y \subseteq V(C) \setminus \{v\}$ tel que $q_{C-v}(Y) > |Y|$. Par la proposition 10.11, $q_{C-v}(Y) - |Y|$ doit être pair et donc

$$q_{C-v}(Y) \ \geq \ |Y| + 2.$$

Puisque X, Y et $\{v\}$ sont deux à deux disjoints,

$$\begin{aligned} q_G(X \cup Y \cup \{v\}) &= q_G(X) - 1 + q_C(Y \cup \{v\}) \\ &= |X| - 1 + q_{C-v}(Y) \\ &\geq |X| - 1 + |Y| + 2 \\ &= |X \cup Y \cup \{v\}|. \end{aligned}$$

Donc $X \cup Y \cup \{v\}$ est une barrière, ce qui contredit la maximalité de X.

Soit G' le graphe biparti avec bipartition $V(G') = X \dot\cup Z$ obtenu en supprimant les arêtes ayant leurs deux extrémités dans X et en contractant chaque composante connexe impaire de $G-X$ en un sommet (Z est l'ensemble des sommets contractés).

Montrons que G' a un couplage parfait. Dans le cas contraire, il existe par le théorème de Frobenius 10.4, $A \subseteq Z$ tel que $|\Gamma_{G'}(A)| < |A|$. Cela implique $q_G(\Gamma_{G'}(A)) \geq |A| > |\Gamma_{G'}(A)|$, menant à une contradiction. □

La preuve précédente est due à Anderson [1971]. La condition de Tutte fournit une bonne caractérisation pour le problème du couplage parfait : soit un graphe possède un couplage parfait, soit il possède un **ensemble de Tutte** X certifiant qu'il n'a pas de couplage parfait. Une importante conséquence du théorème de Tutte est la formule de Berge-Tutte :

Théorème 10.14. (Berge [1958])

$$2\nu(G) + \max_{X \subseteq V(G)} (q_G(X) - |X|) \ = \ |V(G)|.$$

Preuve. Pour tout $X \subseteq V(G)$, tout couplage doit laisser au moins $q_G(X) - |X|$ sommets non couverts. Donc $2\nu(G) + q_G(X) - |X| \leq |V(G)|$.

Pour montrer l'inverse, soit

$$k \ := \ \max_{X \subseteq V(G)} (q_G(X) - |X|).$$

Construisons un nouveau graphe H en ajoutant k nouveaux sommets à G, chacun d'entre eux étant connecté à tous les anciens sommets.

Si nous montrons que H a un couplage parfait, alors

$$2\nu(G) + k \ \geq \ 2\nu(H) - k \ = \ |V(H)| - k \ = \ |V(G)|,$$

et le théorème sera démontré.

Supposons que H n'ait pas de couplage parfait ; alors, par le théorème de Tutte, il existe $Y \subseteq V(H)$ tel que $q_H(Y) > |Y|$. Par la proposition 10.11, k a la même parité que $|V(G)|$, ce qui implique que $|V(H)|$ est pair. Donc $Y \neq \emptyset$ et $q_H(Y) > 1$. Mais Y contient tous les nouveaux sommets et

$$q_G(Y \cap V(G)) \;=\; q_H(Y) \;>\; |Y| \;=\; |Y \cap V(G)| + k,$$

contredisant la définition de k. □

Terminons ce paragraphe avec une proposition que nous utiliserons ultérieurement.

Proposition 10.15. *Soit G un graphe et $X \subseteq V(G)$ tel que $|V(G)| - 2\nu(G) = q_G(X) - |X|$. Tout couplage maximum de G contient un couplage parfait dans chaque composante connexe paire de $G - X$, contient un couplage presque parfait dans chaque composante connexe impaire de $G - X$, et couple tous les sommets de X à des sommets de composantes connexes impaires distinctes de $G - X$.* □

Nous verrons plus tard (théorème de 10.32) qu'on peut choisir X de telle sorte que chaque composante connexe impaire de $G - X$ soit facteur-critique.

10.4 Décompositions en oreilles des graphes facteur-critiques

Ce paragraphe présente quelques résultats sur les graphes facteur-critiques qui seront utiles par la suite. L'exercice 17 du chapitre 2 nous a montré que les graphes qui ont une décomposition en oreilles sont les graphes 2-arête-connexes. Nous serons ici intéressés par les décompositions en oreilles impaires.

Définition 10.16. *Une décomposition en oreilles est appelée* **décomposition en oreilles impaires** *si chaque oreille a une longueur impaire.*

Théorème 10.17. (Lovász [1972]) *Un graphe est facteur-critique si et seulement s'il admet une décomposition en oreilles impaires. De plus, le sommet initial de la décomposition peut être choisi arbitrairement.*

Preuve. Soit G un graphe ayant une décomposition en oreilles impaires. Montrons que G est facteur-critique par induction sur le nombre d'oreilles. Soit P la dernière oreille ; P est une chaîne entre deux sommets x et y ; soit G' le graphe avant l'addition de P. Montrons que $G - v$ contient un couplage parfait pour tout $v \in V(G)$. Cela est évident par induction si v n'est pas un sommet interne de P (ajouter les arêtes d'indice pair de P au couplage de $G' - v$). Si v est un sommet interne de P, alors exactement une des chaînes $P_{[v,x]}$, $P_{[v,y]}$ est paire, $P_{[v,x]}$ par exemple. Par induction, $G' - x$ a un couplage parfait. En ajoutant les arêtes d'indice pair de $P_{[y,v]}$ et d'indice pair de $P_{[v,x]}$ au couplage de $G' - v$, nous obtenons un couplage parfait de $G - v$.

Montrons maintenant la réciproque. Choisissons un sommet initial quelconque z de la décomposition en oreilles, et soit M un couplage presque parfait de G couvrant $V(G)\backslash\{z\}$. Supposons qu'il existe déjà une décomposition en oreilles impaires d'un sous-graphe G' de G tel que $z \in V(G')$ et $M \cap E(G')$ est un couplage presque parfait de G'. Si $G = G'$, la preuve est terminée.

Si G' est différent de G, il existe – puisque G est connexe – une arête $e = (x, y) \in E(G) \setminus E(G')$ avec $x \in V(G')$. Si $y \in V(G')$, e est l'oreille suivante. Sinon soit N un couplage presque parfait de G couvrant $V(G)\backslash\{y\}$. $M \triangle N$ contient évidemment une chaîne P de y à z. Soit w le premier sommet de P (quand on part de y) qui appartient à $V(G')$. La dernière arête de $P' := P_{[y,w]}$ ne peut pas appartenir à M (puisque aucune arête de M ne sort de $V(G')$), et la première arête n'appartient pas à N. Puisque P' est une chaîne M-N-alternée, $|E(P')|$ est pair ; en ajoutant e à P', on obtient l'oreille suivante impaire. □

Définition 10.18. *Soit G un graphe facteur-critique ayant un couplage presque parfait M ; une* **décomposition en oreilles *M*-alternées** *de G est une décomposition en oreilles impaires telle que chaque oreille soit une chaîne M-alternée ou un cycle C avec $|E(C) \cap M| + 1 = |E(C) \setminus M|$.*

Il est évident que le sommet initial d'une décomposition en oreilles M-alternées doit être un sommet non couvert par M. La preuve du théorème 10.17 conduit à :

Corollaire 10.19. *Si G est un graphe facteur-critique et M est un couplage presque parfait de G, il existe une décomposition en oreilles M-alternées.* □

Par la suite, nous serons seulement intéressés par les décompositions en oreilles M-alternées. Une manière intéressante de représenter ce type de décompositions a été proposée par Lovász et Plummer [1986] et se décrit de la manière suivante :

Définition 10.20. *Soit G un graphe facteur-critique et M un couplage presque parfait de G. Soit $r, P_1, \ldots, P_k$ une décomposition en oreilles M-alternées de G et $\mu, \varphi : V(G) \to V(G)$ deux fonctions. Nous dirons que μ et φ sont* **associées à la décomposition en oreilles** *$r, P_1, \ldots, P_k$ si :*

- *$\mu(x) = y$ si $(x, y) \in M$;*
- *$\varphi(x) = y$ si $(x, y) \in E(P_i) \setminus M$ et $x \notin \{r\} \cup V(P_1) \cup \cdots \cup V(P_{i-1})$;*
- *$\mu(r) = \varphi(r) = r$.*

Si M est fixé, nous dirons aussi que φ est associé à $r, P_1, \ldots, P_k$.

Si M est un couplage presque parfait et μ, φ sont associées à deux décompositions en oreilles M-alternées, elles sont identiques à l'ordre près des oreilles. De plus une liste explicite des oreilles peut être obtenue en temps linéaire :

ALGORITHME DE DÉCOMPOSITION EN OREILLES	
Input	Un graphe facteur-critique G, des fonctions μ, φ associées à une décomposition en oreilles M-alternées.
Output	Une décomposition en oreilles M-alternées $r, P_1, \ldots, P_k$.

① Initialement $X := \{r\}$, où r est le sommet avec $\mu(r) = r$.
$k := 0$, la pile est vide.

② **If** $X = V(G)$ **then go to** ⑤.
If la pile n'est pas vide
then soit $v \in V(G) \setminus X$ une extrémité du plus haut élément de la pile,
else choisir $v \in V(G) \setminus X$ arbitrairement.

③ $x := v$, $y := \mu(v)$ et $P := (\{x, y\}, \{(x, y)\})$.
While $\varphi(\varphi(x)) = x$ **do** :
$P := P + (x, \varphi(x)) + (\varphi(x), \mu(\varphi(x)))$ et $x := \mu(\varphi(x))$.
While $\varphi(\varphi(y)) = y$ **do** :
$P := P + (y, \varphi(y)) + (\varphi(y), \mu(\varphi(y)))$ et $y := \mu(\varphi(y))$.
$P := P + (x, \varphi(x)) + (y, \varphi(y))$. P est l'oreille contenant y comme sommet interne. Placer P au plus haut dans la pile.

④ **While** les deux extrémités du plus haut élément de la pile P sont dans X **do** :
Supprimer P de la pile, $k := k + 1$, $P_k := P$ et $X := X \cup V(P)$.
Go to ②.

⑤ **For** tout $(y, z) \in E(G) \setminus (E(P_1) \cup \cdots \cup E(P_k))$ **do** :
$k := k + 1$ et $P_k := (\{y, z\}, \{(y, z)\})$.

Proposition 10.21. *Soit G un graphe facteur-critique et μ, φ des fonctions associées à une décomposition en oreilles M-alternées. Cette décomposition est unique à l'ordre des oreilles près. L'*ALGORITHME DE DÉCOMPOSITION EN OREILLES *fournit explicitement une liste de ces oreilles ; il s'exécute en temps linéaire.*

Preuve. Soit $\mathcal{D}$ une décomposition en oreilles M-alternées associée à μ et φ. L'unicité de $\mathcal{D}$ est la conséquence évidente du fait que P calculé comme dans ③ est bien une oreille de $\mathcal{D}$. Le temps de calcul ① – ④ est clairement $O(|V(G)|)$, tandis que ⑤ nécessite un temps $O(|E(G)|)$. □

La propriété la plus importante est maintenant la suivante :

Lemme 10.22. *Soit G un graphe facteur-critique et soient μ, φ deux fonctions associées à une décomposition en oreilles M-alternées. Soit r un sommet non couvert par M. Alors la chaîne maximale produite par une sous-suite initiale de*

$$x, \mu(x), \varphi(\mu(x)), \mu(\varphi(\mu(x))), \varphi(\mu(\varphi(\mu(x)))), \ldots$$

est une chaîne M-alternée de x à r de longueur paire pour tout $x \in V(G)$.

Preuve. Soit $x \in V(G) \setminus \{r\}$, et soit P_i la première oreille contenant x. Clairement une sous-suite initiale de

$$x, \mu(x), \varphi(\mu(x)), \mu(\varphi(\mu(x))), \varphi(\mu(\varphi(\mu(x)))), \ldots$$

est une sous-chaîne Q de P_i allant de x à y, où $y \in \{r\} \cup V(P_1) \cup \cdots \cup V(P_{i-1})$. Comme nous avons une décomposition M-alternée, la dernière arête de Q n'appartient pas à M ; donc Q a une longueur paire. Si $y = r$, le lemme est démontré, sinon nous faisons une induction sur i. □

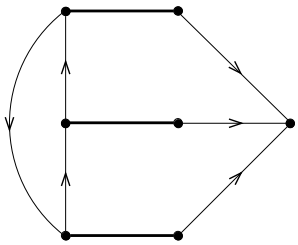


Figure 10.2.

Le contraire du lemme 10.22 n'est pas vrai : dans le contre-exemple décrit sur la figure 10.2 (les arêtes en gras sont dans le couplage, les arêtes orientées de u vers v indiquent que $\varphi(u) = v$), μ et φ induisent des chaînes alternées vers le sommet non couvert par le couplage. Cependant, μ et φ ne sont pas associées à une quelconque décomposition en oreilles alternées.

Dans l'ALGORITHME DE COUPLAGE DE POIDS MAXIMUM (paragraphe 11.3), nous aurons besoin d'une procédure rapide pour mettre à jour une décomposition en oreilles alternées quand le couplage change. Bien que la preuve du théorème 10.17 soit algorithmique (pourvu que l'on sache trouver un couplage maximum), la procédure induite par cette preuve est par trop inefficace ; aussi aurons-nous besoin de la décomposition en oreilles impaires :

Lemme 10.23. *Soient G un graphe facteur-critique, deux couplages presque parfaits M et M', et des fonctions μ, φ associées à une décomposition en oreilles M-alternées. Alors les fonctions μ', φ' associées à une décomposition en oreilles M'-alternées peuvent être trouvées en un temps $O(|V(G)|)$.*

Preuve. Soient v le sommet non couvert par M et v' le sommet non couvert par M'. Soit $P = x_0, x_1, \ldots, x_k$ la chaîne de v' à v dans $M \triangle M'$ ($x_0 = v'$ et $x_k = v$).

Une liste explicite de la première décomposition en oreilles peut se trouver à partir de μ et φ par l'ALGORITHME DE DÉCOMPOSITION EN OREILLES en temps linéaire (proposition 10.21). En fait, puisque nous n'avons pas à considérer les oreilles de longueur un, nous pouvons oublier ⑤ : donc le nombre total d'arêtes considérées est au plus $\frac{3}{2}(|V(G)| - 1)$ (voir exercice 19).

Supposons que nous ayons déjà construit une décomposition en oreilles M'-alternées d'un sous-graphe couvrant $G[X]$ pour $X \subseteq V(G)$ avec $v' \in X$ (initialement $X := \{v'\}$). Bien sûr, aucune arête de M' ne sort de X. Soit $p := \max\{i \in \{0, \ldots, k\} : x_i \in X\}$ (illustration sur la figure 10.3). À chaque étape nous gardons

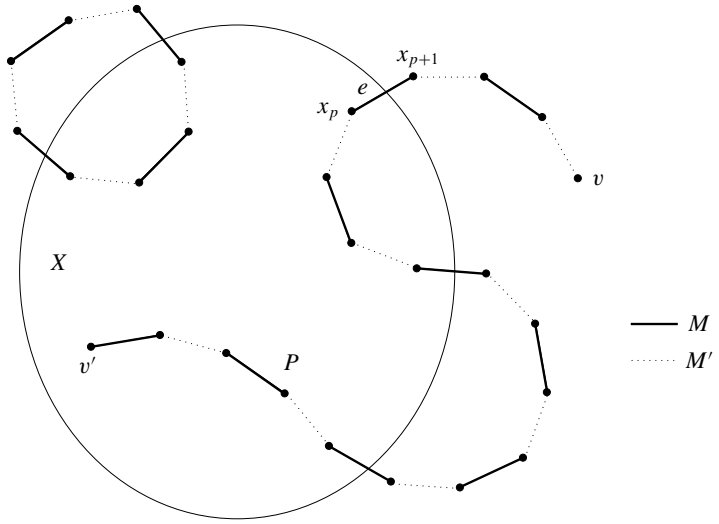


Figure 10.3.

trace de p et de l'ensemble d'arêtes $\delta(X) \cap M$. Leur mise à jour quand X croît est clairement possible en temps linéaire.

Montrons maintenant comment étendre la décomposition en oreilles. À chaque étape nous ajouterons au moins une oreille, le temps nécessaire étant proportionnel au nombre total d'arêtes dans les nouvelles oreilles.

Cas 1 : $|\delta(X) \cap M| \geq 2$. Soit $f \in \delta(X) \cap M$ avec $x_p \notin f$. Manifestement, f appartient à une chaîne M-M'-alternée qui peut être rajoutée comme oreille suivante Le temps nécessaire pour trouver cette oreille est proportionnel à sa longueur.

Cas 2 : $|\delta(X) \cap M| = 1$. Alors $v \notin X$, et $e = (x_p, x_{p+1})$ est la seule arête de $\delta(X) \cap M$. Soit R' la chaîne de x_{p+1} à v définie par μ et φ (voir lemme 10.22). La première arête de R' est e. Soit q l'indice minimum $i \in \{p+2, p+4, \ldots, k\}$ tel que $x_i \in V(R')$ et $V(R'_{[x_{p+1}, x_i]}) \cap \{x_{i+1}, \ldots, x_k\} = \emptyset$ (voir figure 10.4). Soit $R := R'_{[x_p, x_q]}$. R a pour sommets x_p, $\varphi(x_p)$, $\mu(\varphi(x_p))$, $\varphi(\mu(\varphi(x_p)))$, $\ldots, x_q$, et peut être parcouru en un temps proportionnel à sa longueur.

Soit $S := E(R) \setminus E(G[X])$, $D := (M \triangle M') \setminus (E(G[X]) \cup E(P_{[x_q, v]}))$, et soit $Z := S \triangle D$. S et D sont des chaînes et des cycles M-alternés. Observons que chaque sommet non dans X a degré 0 ou 2 par rapport à Z. De plus, si un sommet non dans X a deux arêtes incidentes dans Z, une des deux appartient à M'. (Le choix de q est essentiel ici.)

Il s'ensuit que toutes les composantes connexes C de $(V(G), Z)$ avec $E(C) \cap \delta(X) \neq \emptyset$ peuvent être ajoutées comme oreilles suivantes, et, après cette extension, $S \setminus Z = S \cap (M \triangle M')$ est l'union de chaînes sommet-disjointes, chacune d'entre elles pouvant être ajoutée comme nouvelle oreille. Comme $e \in D \setminus S \subseteq Z$, nous avons $Z \cap \delta(X) \neq \emptyset$, et au moins une oreille est ajoutée.

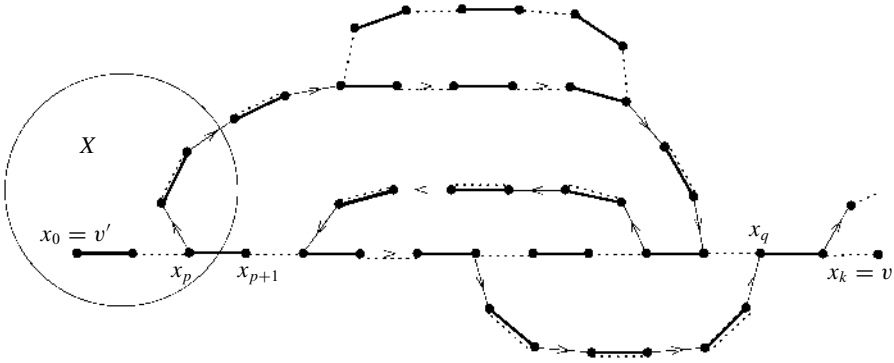


Figure 10.4.

Il reste à montrer que le temps nécessaire à la construction précédente est proportionnel au nombre total d'arêtes dans les nouvelles oreilles. Il est évident qu'il suffit de trouver S en un temps $O(|E(S)|)$.

Cela est difficile à cause des sous-chaînes de R qui sont dans X. Cependant il n'est pas indispensable de connaître ces sous-chaînes. Nous devons pouvoir les raccourcir quand c'est possible. À cet effet, nous modifierons les variables φ.

Dans chaque application du cas 2, soit $R_{[a,b]}$ une sous-chaîne maximale de R à l'intérieur de X avec $a \neq b$. Soit $y := \mu(b)$; y est le prédécesseur de b sur R. Posons $\varphi(x) := y$ pour les sommets x sur $R_{[a,y]}$ tels que $R_{[x,y]}$ a une longueur impaire. x et y peuvent être ou ne pas être adjacents (voir figure 10.5).

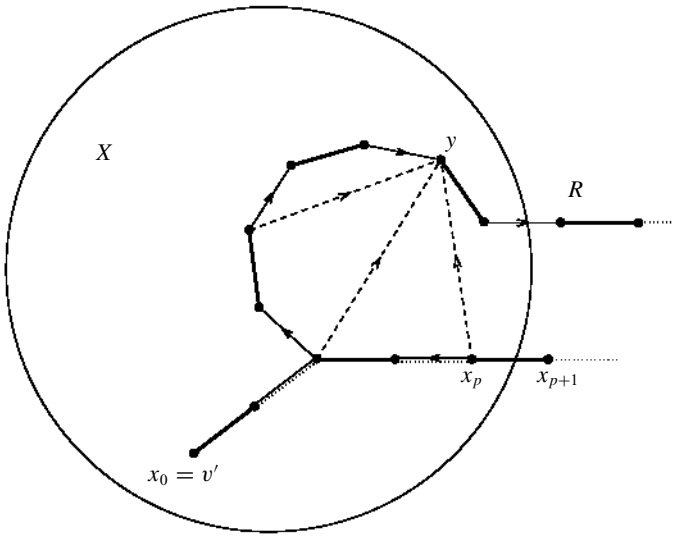


Figure 10.5.

Le temps nécessaire pour mettre à jour les variables φ est proportionnel au nombre d'arêtes examinées. Remarquons que les changements de φ ne modifient pas la propriété du lemme 10.22, et que les variables φ ne sont plus utilisées sauf pour trouver des chaînes M-alternées vers v dans le cas 2.

Il est alors certain que le temps nécessaire pour trouver les sous-chaînes de R qui sont dans X est proportionnel au nombre de sous-chaînes auquel s'ajoute le nombre d'arêtes examinées pour la première fois à l'intérieur de X. Puisque le nombre de sous-chaînes dans X est plus petit ou égal au nombre de nouvelles oreilles à cette étape, le temps total de calcul est linéaire.

Cas 3 : $\delta(X) \cap M = \emptyset$. Alors $v \in X$. Soit R la première oreille de la décomposition en oreilles M-alternées telle que $V(R) \setminus X \neq \emptyset$.

Comme dans le cas 2, soient $S := E(R) \setminus E(G[X])$, $D := (M \triangle M') \setminus E(G[X])$, et $Z := S \triangle D$. Ici encore, les composantes connexes C de $(V(G), Z)$ avec $E(C) \cap \delta(X) \neq \emptyset$ peuvent être ajoutées comme oreilles suivantes, et après cela, $S \setminus Z$ est l'union de chaînes arête-disjointes, chacune d'entre elles pouvant être ajoutée comme oreille. Le temps total de calcul pour le cas 3 est linéaire. □

10.5 Algorithme du couplage d'Edmonds

Par le théorème de Berge 10.7, un couplage est maximum si et seulement s'il n'existe pas de chaîne alternée augmentante. Notre algorithme de couplage sera donc fondé sur l'existence et la détection de chaînes alternées augmentantes.

Il n'est cependant pas évident de trouver une chaîne alternée augmentante (ou de montrer qu'il n'y en a pas). Dans le cas biparti (théorème 10.5) il était suffisant de marquer les sommets x extrémités d'une chaîne alternée de v à x partant de v, sommet non couvert par le couplage. Puisqu'il n'existe pas de cycles impairs, cette procédure se fait sans ambiguité, ce qui n'est pas le cas en général.

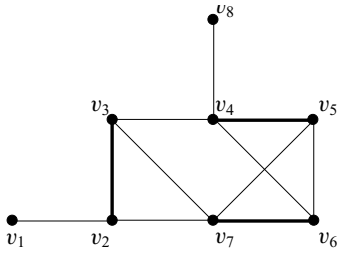


Figure 10.6.

Considérons l'exemple de la figure 10.6 (les arêtes en gras constituent un couplage M). En partant de v_1, nous obtenons un parcours alterné $v_1, v_2, v_3, v_4, v_5, v_6$, v_7, v_5, v_4, v_8. Ce parcours contient un cycle impair v_5, v_6, v_7. Notons que, dans cet

exemple, il existe une chaîne alternée augmentante $(v_1, v_2, v_3, v_7, v_6, v_5, v_4, v_8)$, mais il n'y a pas de méthode apparente pour la trouver.

Nous verrons dans le lemme 10.25 que, quand un cycle impair est détecté, il suffit de le contracter en un seul sommet pour l'éliminer ; le nouveau graphe a un couplage parfait si et seulement si le graphe original a un couplage parfait. C'est l'idée générale de l'ALGORITHME DE COUPLAGE MAXIMUM D'EDMONDS. Donnons d'abord la définition suivante :

Définition 10.24. *Soit G un graphe ayant un couplage M. Un* **blossom**[1] *par rapport à M est un sous-graphe facteur-critique C de G tel que $|M \cap E(C)| = \frac{|V(C)|-1}{2}$. Le sommet de C non couvert par $M \cap E(C)$ est appelé* **base** *de C.*

Le blossom que nous avons rencontré dans l'exemple ci-dessus (figure 10.6) est induit par $\{v_5, v_6, v_7\}$. Notons que cet exemple contient d'autres blossoms. D'autre part, notre définition implique que tout sommet est un blossom. Nous pouvons formuler le lemme du blossom contracté :

Lemme 10.25. *Soit G un graphe ayant un couplage M, et soit C un blossom dans G (par rapport à M). Supposons qu'il existe une chaîne M-alternée Q de v à r de longueur paire connectant un sommet v non couvert par M à la base r de C, avec $E(Q) \cap E(C) = \emptyset$.*

Soient G' et M' se déduisant de G et M par contraction de $V(C)$ en un seul sommet. Alors M est un couplage maximum de G si et seulement si M' est un couplage maximum de G'.

Preuve. Supposons que M ne soit pas un couplage maximum de G. $N := M \triangle E(Q)$ est un couplage de même cardinalité qui n'est pas non plus maximum. Par le théorème de Berge 10.7, il existe une chaîne N-augmentante P dans G. Remarquons que N ne couvre pas r.

Au moins une des deux extrémités de P, par exemple x, n'est pas dans C. Si P et C sont disjoints, soit y l'autre extrémité de P. Sinon soit y le premier sommet de P – quand on décrit P à partir de x – qui appartient à C. Soit P' se déduisant de $P_{[x,y]}$ après contraction de $V(C)$ dans G. Les extrémités de P' ne sont pas couvertes par N' (le couplage dans G' correspondant à N). Donc P' est une chaîne N'-augmentante de G'. Donc N' n'est pas un couplage maximum de G', et M', qui a la même cardinalité, non plus.

Pour montrer la réciproque, supposons que M' ne soit pas un couplage maximum de G'. Soit N' un couplage de G' de taille plus grande. N' correspond à un couplage N_0 de G qui couvre au plus un sommet de C dans G. Puisque C est facteur-critique, on peut ajouter à N_0 $k := \frac{|V(C)|-1}{2}$ arêtes afin d'obtenir un couplage N de G, tel que

$$|N| = |N_0| + k = |N'| + k > |M'| + k = |M|,$$

ce qui montre que M n'est pas un couplage maximum de G. □

[1] Nous garderons ici le terme anglais original (*ndt*).

Il est nécessaire d'imposer que la base du blossom soit joignable d'un sommet non couvert par M par une chaîne M-alternée de longueur paire disjointe du blossom. Par exemple, le blossom induit par $\{v_4, v_6, v_7, v_2, v_3\}$ sur la figure 10.6 ne peut être contracté sans détruire la seule chaîne augmentante.

Construisons maintenant une forêt augmentante :

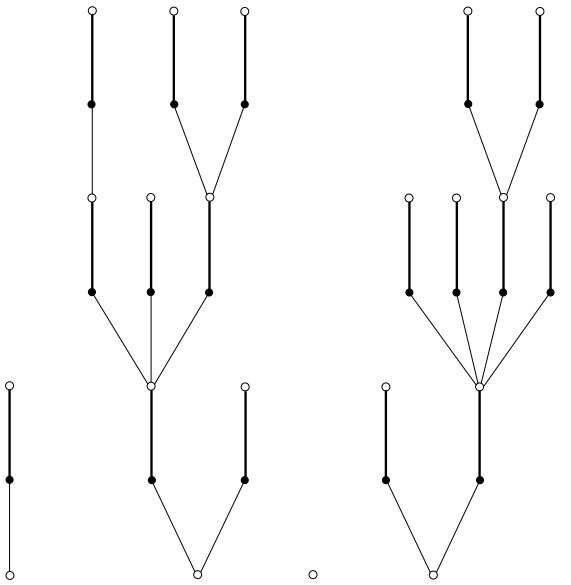


Figure 10.7.

Définition 10.26. *Soit un graphe G et un couplage M de G. Une* **forêt alternée** *par rapport à M dans G est une forêt F de G ayant les propriétés suivantes :*

(a) *$V(F)$ contient tous les sommets non couverts par M. Chaque composante connexe de F contient exactement un sommet non couvert par M, sa* **racine**.

(b) *Nous dirons qu'un sommet $v \in V(F)$ est un sommet* **externe (interne)** *s'il est à distance paire (impaire) de la racine de la composante connexe contenant v. (Les racines, en particulier, sont des sommets externes.) Tous les sommets internes ont degré 2 dans F.*

(c) *Pour tout $v \in V(F)$, la chaîne unique joignant v à la racine de la composante connexe contenant v est M-alternée.*

La figure 10.7 décrit une forêt alternée. Les arêtes en gras sont dans le couplage. Les sommets noirs sont internes, les blancs externes.

Proposition 10.27. *Dans toute forêt alternée le nombre de sommets externes distincts de la racine est égal au nombre de sommets internes.*

Preuve. Chaque sommet externe distinct de la racine a exactement un voisin dont la distance à la racine est plus petite. Cela à l'évidence établit une bijection entre les sommets externes distincts de la racine et les sommets internes. □

De manière informelle, l'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS se déroule de la manière suivante. Étant donné un couplage M, construisons une forêt M-alternée F, en commençant avec l'ensemble S des sommets non couverts par M, et aucune arête.

À chaque étape de l'algorithme, choisissons un voisin y d'un sommet externe x. Soit $P(x)$ l'unique chaîne F de x à la racine. Il y a trois cas intéressants qui correspondent aux trois opérations («grossir», «augmenter», et «contracter») :

Cas 1 : $y \notin V(F)$. Alors la forêt grossit quand nous lui ajoutons (x, y) et l'arête du couplage couvrant y.

Cas 2 : y est un sommet externe situé dans une composante connexe différente de F. Alors nous augmentons M le long de $P(x) \cup (x, y) \cup P(y)$.

Cas 3 : y est un sommet externe dans la même composante connexe de F (la racine étant q). Soit r le premier sommet de $P(x)$ (en partant de x) qui appartient à $P(y)$. (r peut être un des deux sommets x, y.) Si r n'est pas une racine, il doit avoir degré au moins 3 et r est un sommet externe. Donc $C := P(x)_{[x,r]} \cup (x, y) \cup P(y)_{[y,r]}$ est un blossom avec au moins trois sommets. Nous contractons C.

Si aucun de ces cas ne s'applique, tous les voisins des sommets externes sont internes. Montrons que M est alors maximum. Soit X l'ensemble des sommets internes, $s := |X|$, et soit t le nombre de sommets externes. $G - X$ a t composantes connexes impaires (chaque sommet externe est isolé dans $G-X$), et $q_G(X)-|X| = t - s$. Donc par la partie triviale de la formule de Berge-Tutte, tout couplage laisse au moins $t - s$ sommets non couverts. Mais le nombre de sommets non couverts par M, c.-à-d. le nombre de racines de F, est précisément $t - s$ par la proposition 10.27. Donc M est bien maximum.

Nous étudierons maintenant les questions d'implémentations algorithmiques qui ne sont pas triviales. Le problème est de savoir comment traiter les opérations de contraction de manière efficace afin de pouvoir retrouver le graphe original après la contraction. Bien entendu, un même sommet peut être concerné par plusieurs opérations de contraction successives. Notre présentation est fondée sur celle donnée par Lovász et Plummer [1986].

Plutôt que d'implémenter les opérations de contraction, nous autoriserons nos forêts à avoir des blossoms.

Définition 10.28. *Soit un graphe G ayant un couplage M. Un sous-graphe F de G est une* **forêt générale de blossoms** *(par rapport à M) s'il existe une partition $V(F) = V_1 \dot\cup V_2 \dot\cup \cdots \dot\cup V_k$ de l'ensemble des sommets telle que $F_i := F[V_i]$ soit un sous-graphe facteur-critique maximal de F avec $|M \cap E(F_i)| = \frac{|V_i|-1}{2}$ ($i = 1, \ldots, k$) et qu'après contraction de chaque $V_1, \ldots, V_k$, on obtienne une forêt alternée F'.*

F_i est appelé **blossom externe (blossom interne)** *si V_i est un sommet externe (interne) dans F'. Tous les sommets d'un blossom externe (interne) sont appelés* **ex-**

ternes (internes). *Une forêt de blossoms générale telle que chaque blossom interne soit réduit à un seul sommet sera appelée* **forêt spéciale de blossoms**.

La figure 10.8 montre une composante connexe d'une forêt spéciale de blossoms avec cinq blossoms externes non triviaux. Elle correspond à une des composantes connexes de la forêt alternée de la figure 10.7. Les orientations données aux arêtes seront expliquées plus tard. Les sommets de G qui n'appartiennent pas à une forêt spéciale de blossoms seront dits **hors de la forêt**.

Notons que le lemme du blossom contracté 10.25 est valable uniquement pour les blossoms externes. Dans ce chapitre, contrairement au suivant, où apparaîtront les forêts générales de blossoms, nous ne serons concernés que par les forêts spéciales de blossoms.

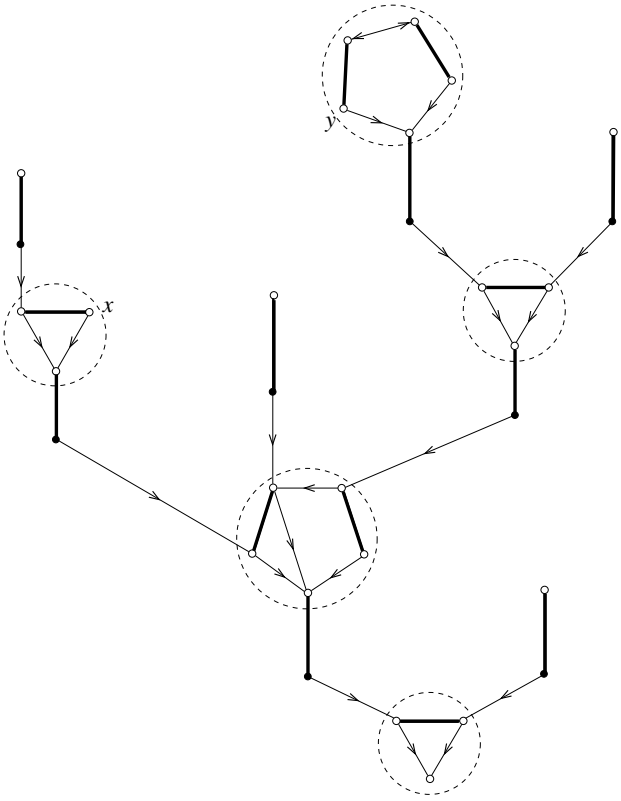


Figure 10.8.

Afin de mémoriser une forêt spéciale de blossoms F, nous introduirons les structures de données suivantes : nous associerons trois variables $\mu(x)$, $\varphi(x)$, et $\rho(x)$ à chaque sommet $x \in V(G)$ ayant les propriétés suivantes :

$$\mu(x) = \begin{cases} x & \text{si } x \text{ n'est pas couvert par } M \\ y & \text{si } (x,y) \in M \end{cases} \tag{10.2}$$

$$\varphi(x) = \begin{cases} x & \text{si } x \notin V(F) \text{ ou } x \text{ est la base d'un blossom externe} \\ y & \text{pour } (x,y) \in E(F) \setminus M \text{ si } x \text{ est un sommet interne} \\ y & \text{tel que } (x,y) \in E(F) \setminus M, \text{ et } \mu \text{ et } \varphi \text{ sont} \\ & \text{associés à une décomposition en oreilles } M\text{-alternée} \\ & \text{du blossom contenant } x, \text{ si } x \text{ est un sommet externe} \end{cases} \tag{10.3}$$

$$\rho(x) = \begin{cases} x & \text{si } x \text{ n'est pas un sommet externe} \\ y & \text{si } x \text{ est un sommet externe et } y \text{ est la base du} \\ & \text{blossom externe de } F \text{ contenant } x. \end{cases} \tag{10.4}$$

Pour chaque sommet externe v, nous définissons $P(v)$ comme étant la chaîne maximale donnée par une sous-suite initiale de

$$v, \mu(v), \varphi(\mu(v)), \mu(\varphi(\mu(v))), \varphi(\mu(\varphi(\mu(v)))), \ldots$$

Nous avons les propriétés suivantes :

Proposition 10.29. *Soit F une forêt spéciale de blossoms par rapport à un couplage M, et soient $\mu, \varphi : V(G) \to V(G)$ des fonctions vérifiant (10.2) et (10.3). Nous avons alors :*

(a) *Pour chaque sommet externe v, $P(v)$ est une chaîne alternée de v à q, q étant la racine de l'arbre de F contenant v.*

(b) *Un sommet x est :*
- *externe si et seulement si $\mu(x) = x$ ou $\varphi(\mu(x)) \neq \mu(x)$;*
- *interne si et seulement si $\varphi(\mu(x)) = \mu(x)$ et $\varphi(x) \neq x$;*
- *hors de la forêt si et seulement si $\mu(x) \neq x$, $\varphi(x) = x$ et $\varphi(\mu(x)) = \mu(x)$.*

Preuve. (a) : par (10.3) et le lemme 10.22, il existe une suite

$$v, \mu(v), \varphi(\mu(v)), \mu(\varphi(\mu(v))), \varphi(\mu(\varphi(\mu(v)))), \ldots$$

qui est une chaîne M-alternée de longueur paire joignant v à la base r du blossom contenant v. Si r n'est pas la racine de l'arbre contenant v, r est couvert par M. Donc la suite précédente se poursuit par l'introduction de l'arête du couplage $(r, \mu(r))$ et aussi de l'arête $\mu(r), \varphi(\mu(r))$, parce que $\mu(r)$ est un sommet interne. Mais $\varphi(\mu(r))$ est un sommet externe et la condition (a) est donc vraie par induction.

(b) : si le sommet x est externe, soit x est une racine (c.-à-d. $\mu(x) = x$), soit $P(x)$ est une chaîne de longueur au moins deux, c.-à-d. $\varphi(\mu(x)) \neq \mu(x)$.

Si x est interne, $\mu(x)$ est la base d'un blossom externe ; par (10.3) $\varphi(\mu(x)) = \mu(x)$. Comme $P(\mu(x))$ est une chaîne de longueur au moins 2, $\varphi(x) \neq x$.

Si x est hors de la forêt, par définition x est couvert par M, donc par (10.2) $\mu(x) \neq x$. $\mu(x)$ est aussi hors de la forêt ; par (10.3) $\varphi(x) = x$ et $\varphi(\mu(x)) = \mu(x)$.

Puisque tout sommet est externe, interne ou hors de la forêt et puisque tout sommet vérifie une des trois conditions de (b), la proposition est démontrée. □

Sur la figure 10.8, une arête est orientée de u vers v si $\varphi(u) = v$. Nous sommes maintenant prêts pour une description précise de l'algorithme.

ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS

Input Un graphe G.

Output Un couplage de cardinalité maximum G décrit par ses arêtes $(x, \mu(x))$.

① $\mu(v) := v$, $\varphi(v) := v$, $\rho(v) := v$ et $balayage(v) :=$ *faux* pour tout $v \in V(G)$.

② **If** tous les sommets externes sont balayés (scannés)
then stop,
else soit x un sommet externe avec $balayage(x) =$ *faux*.

③ Soit y un voisin de x tel que y soit hors de la forêt ou (y est externe et $\rho(y) \neq \rho(x)$).
If aucun y n'existe **then** $balayage(x) :=$ *vrai* et **go to** ②.

④ («grossir»)
If y est hors de la forêt **then** $\varphi(y) := x$ et **go to** ③.

⑤ («augmenter»)
If $P(x)$ et $P(y)$ sont sommet-disjoints **then**
$\mu(\varphi(v)) := v$, $\mu(v) := \varphi(v)$ pour tout $v \in V(P(x)) \cup V(P(y))$ à distance impaire de x sur $P(x)$ ou y sur $P(y)$.
$\mu(x) := y$.
$\mu(y) := x$.
$\varphi(v) := v$, $\rho(v) := v$, $balayage(v) :=$ *faux* pour tout $v \in V(G)$.
Go to ②.

⑥ («contracter»)
Soit r le premier sommet sur $V(P(x)) \cap V(P(y))$ avec $\rho(r) = r$.
For $v \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$ à distance impaire de x ou y sur, respectivement, $P(x)_{[x,r]}$ ou $P(y)_{[y,r]}$ et $\rho(\varphi(v)) \neq r$ **do** :
$\varphi(\varphi(v)) := v$.
If $\rho(x) \neq r$ **then** $\varphi(x) := y$.
If $\rho(y) \neq r$ **then** $\varphi(y) := x$.
For tout $v \in V(G)$ avec $\rho(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$ **do** :
$\rho(v) := r$.
Go to ③.

Pour une illustration de l'effet de la contraction sur les valeurs φ, voir la figure 10.9, où ⑥ de l'algorithme a été appliqué à x et y sur la figure 10.8.

Lemme 10.30. *À chaque étape de l'*ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS*, les conditions suivantes sont vérifiées :*

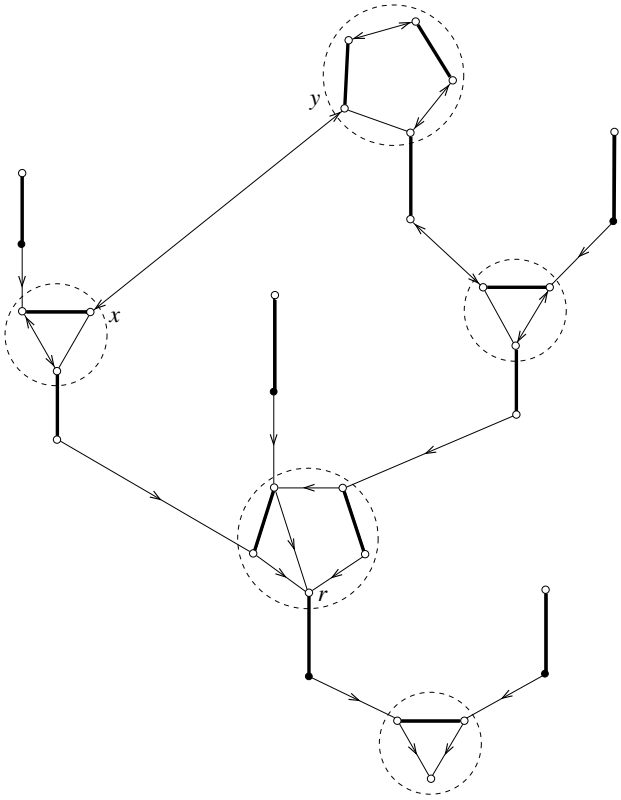


Figure 10.9.

(a) *Les arêtes* $(x, \mu(x))$ *forment un couplage* M.

(b) *Les arêtes* $(x, \mu(x))$ *et* $(x, \varphi(x))$ *forment une forêt spéciale de blossoms* F *par rapport à* M *(avec, en plus, des arêtes isolées du couplage).*

(c) *Le propriétés (10.2), (10.3) et (10.4) sont vérifiées par rapport à* F.

Preuve. (a) : le seul endroit où μ est changé est ⑤, où cette modification est faite manifestement correctement.

(b) : puisque après et ① et ⑤ nous avons une forêt de blossoms sans arêtes et puisque ④ ajoute correctement deux arêtes à la forêt, nous devons seulement vérifier ⑥. r est soit une racine, soit un sommet de degré au moins trois ; il est donc externe. Soit $B := V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$. Soit (u, v) une arête de la forêt de blossoms avec $u \in B$ et $v \notin B$. Puisque $F[B]$ contient un couplage presque parfait, (u, v) est une arête du couplage seulement si c'est l'arête $(r, \mu(r))$. De plus, u a été externe avant d'appliquer ⑥. Cela implique que F continue à être une forêt spéciale de blossoms.

(c) : le seul fait non trivial ici est que, après contraction, μ et φ sont associés à une décomposition en oreilles alternées du nouveau blossom. Soient x et

y deux sommets externes dans la même composante connexe de la forêt spéciale de blossoms, et soit r le premier sommet de $V(P(x)) \cap V(P(y))$ pour lequel $\rho(r) = r$. Le nouveau blossom a pour sommets $B := \{v \in V(G) : \rho(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})\}$.

Nous notons que $\varphi(v)$ n'est pas changé pour tout $v \in B$ avec $\rho(v) = r$. Donc la décomposition en oreilles de l'ancien blossom $B' := \{v \in V(G) : \rho(v) = r\}$ est l'initialisation de la décomposition en oreilles du nouveau B. L'oreille suivante sera $P(x)_{[x,x']}$, $P(y)_{[y,y']}$ avec l'arête (x, y), x' étant le premier sommet de $P(x)$ et y' étant le premier sommet de $P(y)$ appartenant à B'. Finalement, pour chaque oreille Q d'un ancien blossom externe $B'' \subseteq B$, $Q \setminus (E(P(x)) \cup E(P(y)))$ devient une oreille de la nouvelle décomposition en oreilles B. □

Théorème 10.31. (Edmonds [1965]) *L'*ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS *répond correctement en un temps $O(n^3)$ ($n = |V(G)|$).*

Preuve. Le lemme 10.30 et la proposition 10.29 montrent que l'algorithme donne une réponse correcte. Quand l'algorithme se termine avec comme output le couplage M et la forêt spéciale de blossoms F qui vérifient les conditions (a) et (b) du lemme 10.30(a), il est évident que tout voisin d'un sommet externe x est un sommet interne ou un sommet y appartenant au même blossom (c.-à-d. $\rho(y) = \rho(x)$).

Montrons que M est un couplage maximum : soit X l'ensemble des sommets internes, et soit B l'ensemble des sommets qui sont bases d'un blossom externe dans F. Les sommets non couverts appartiennent à B et les sommets couverts de B sont couplés dans M avec les sommets de X :

$$|B| \;=\; |X| + |V(G)| - 2|M|. \tag{10.5}$$

D'autre part, les blossoms externes de F sont les composantes connexes impaires de $G - X$. Donc tout couplage laisse $|B| - |X|$ sommets non couverts. Par (10.5), M laisse exactement $|B| - |X|$ sommets non couverts et est donc maximum.

Analysons le temps de calcul. Par la proposition 10.29(b), le statut de chaque sommet (interne, externe, hors de la forêt) se vérifie en temps constant. ④, ⑤, ⑥ nécessitent un temps $O(n)$. Entre deux augmentations, ④ ou ⑥ sont exécutées au plus $O(n)$ fois, puisque le nombre de points fixes de φ décroît chaque fois. Entre deux augmentations, aucun sommet n'est balayé deux fois. Le temps nécessaire entre deux augmentations est $O(n^2)$ et le temps de calcul total est $O(n^3)$. □

Micali et Vazirani [1980] ont réduit le temps de calcul à $O\left(\sqrt{n}\, m\right)$ en utilisant un résultat de l'exercice 9, mais l'existence de blossoms rend la recherche d'un ensemble maximal de chaînes augmentantes disjointes de longueur minimum plus difficile que dans le cas biparti (cas qui a été résolu précédemment par Hopcroft et Karp [1973], voir exercice 10). Voir également Vazirani [1994]. La meilleure complexité actuelle pour le PROBLÈME DU COUPLAGE MAXIMUM est $O\left(m\sqrt{n}\frac{\log(n^2/m)}{\log n}\right)$, exactement comme dans le cas biparti. Elle a été obtenue par Goldberg et Karzanov [2004] et par Fremuth-Paeger et Jungnickel [2003].

Grâce à l'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS, nous pouvons prouver de manière constructive le théorème de la structure de Gallai-Edmonds. Ce résultat a été montré d'abord par Gallai.

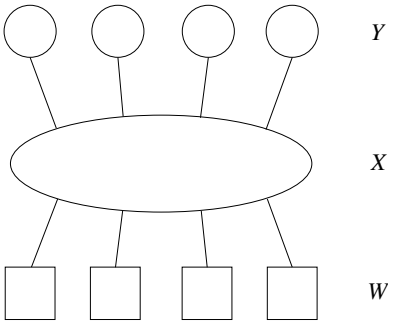


Figure 10.10.

Théorème 10.32. (Gallai [1964]) *Soit G un graphe. Soit Y l'ensemble des sommets non couverts par au moins un couplage maximum, soit X l'ensemble des voisins de Y dans $V(G) \setminus Y$, et soit W l'ensemble des autres sommets. Alors :*

(a) *Tout couplage maximum de G contient un couplage parfait de $G[W]$ et des couplages presque parfaits des composantes connexes de $G[Y]$, et couple tous les sommets de X à des composantes connexes distinctes de $G[Y]$.*

(b) *Les composantes connexes de $G[Y]$ sont facteur-critiques.*

(c) $2\nu(G) = |V(G)| - q_G(X) + |X|$.

W, X, Y sera la **décomposition de Gallai-Edmonds** *de G (voir figure 10.10).*

Preuve. Soit M le couplage et soit F la forêt spéciale de blossoms donnée par l'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS. Soient X' l'ensemble des sommets internes, Y' l'ensemble des sommets externes et W' l'ensemble des sommets hors de la forêt. Montrons d'abord que X', Y', W' vérifient (a)–(c).

La preuve du théorème 10.31 montre que $2\nu(G) = |V(G)| - q_G(X') + |X'|$. Nous appliquons la proposition 10.15 à X'. Puisque les composantes impaires de $G - X'$ sont les blossoms externes de F, (a) est vérifié par X', Y', W'. Puisque les blossoms externes sont facteur-critiques, (b) est également vrai.

Puisque (a) est vérifié par X', Y' et W', tout couplage maximum couvre tous les sommets de $V(G) \setminus Y'$. On a donc $Y \subseteq Y'$. Montrons que l'on a également $Y' \subseteq Y$. Soit v un sommet externe de F. Alors $M \triangle E(P(v))$ est un couplage maximum M', et M' ne couvre pas v. Donc $v \in Y$. Par conséquent $Y = Y'$, ce qui implique $X = X'$ et $W = W'$. Le théorème est donc prouvé. □

Exercices

1. Soit G un graphe avec deux couplages maximaux M_1, M_2. Montrer que : $|M_1| \leq 2|M_2|$.
2. Soit $\alpha(G)$ la taille d'un ensemble stable maximum de G, et soit $\zeta(G)$ la cardinalité minimum d'une couverture par les arêtes. Montrer :
   (a) $\alpha(G) + \tau(G) = |V(G)|$ pour tout graphe G.
   (b) $\nu(G) + \zeta(G) = |V(G)|$ pour tout graphe G sans sommets isolés.
   (c) $\zeta(G) = \alpha(G)$ pour tout graphe biparti G sans sommets isolés.
   (König [1933], Gallai [1959])
3. Montrer qu'un graphe biparti k-régulier a k couplages parfaits disjoints. En déduire que l'ensemble des arêtes d'un graphe biparti de degré maximum k est partitionnable en k couplages.
   (König [1916] ; voir Rizzi [1998] ou théorème 16.16)

* 4. Un ensemble partiellement ordonné (en anglais *poset*) est un ensemble S muni d'une relation d'ordre, c-.à-d. une relation $R \subseteq S \times S$ réflexive ($(x,x) \in R$ pour tous $x \in S$), antisymétrique (si $(x,y) \in R$ et $(y,x) \in R$ alors $x = y$), et transitive (si $(x,y) \in R$ et $(y,z) \in R$ alors $(x,z) \in R$). Deux éléments $x, y \in S$ seront comparables si $(x,y) \in R$ ou $(y,x) \in R$; sinon ils seront incomparables. Une chaîne (une antichaîne) est un sous-ensemble d'éléments de S comparables (incomparables) deux à deux. Utiliser le théorème de König 10.2 pour montrer le théorème de Dilworth [1950] suivant : dans un ensemble partiellement ordonné, la taille maximum d'une antichaîne est égale au nombre minimum de chaînes qui partitionnent cet ensemble.
   Indication : créer deux copies v' et v'' de chaque $v \in S$ et étudier le graphe avec les arêtes (v', w'') associées aux arcs $(v,w) \in R$.
   (Fulkerson [1956])

5. (a) Soient $S = \{1, 2, \ldots, n\}$ et $0 \leq k < \frac{n}{2}$. Soient A et B les collections de sous-ensembles de taille respectivement k et $(k+1)$ de S. Construire un graphe biparti
   $$G \;=\; (A \,\dot\cup\, B,\; \{\{a,b\} : a \in A,\; b \in B, a \subseteq b\}).$$
   Montrer que G a un couplage couvrant A.

* (b) Montrer le lemme de Sperner : le nombre maximum de sous-ensembles non inclus l'un dans l'autre d'un ensemble de n éléments est $\binom{n}{\lfloor \frac{n}{2} \rfloor}$.
   (Sperner [1928])

6. Soit $(U, \mathcal{S})$ un système d'ensembles. Une application injective $\Phi : \mathcal{S} \to U$ telle que $\Phi(S) \in S$ pour tout $S \in \mathcal{S}$ est appelée système de représentants distincts de $\mathcal{S}$. Montrer :
   (a) $\mathcal{S}$ a un système de représentants distincts si et seulement si l'union de k quelconques de ces ensembles $\mathcal{S}$ a une taille au moins égale à k.
   (Hall [1935])

(b) Pour $u \in U$ soit $r(u) := |\{S \in \mathcal{S} : u \in S\}|$. Soit $n := |\mathcal{S}|$ et $N := \sum_{S\in\mathcal{S}} |S| = \sum_{u\in U} r(u)$. Supposons que $|S| < \frac{N}{n-1}$ pour $S \in \mathcal{S}$ et que $r(u) < \frac{N}{n-1}$ pour $u \in U$. Alors $\mathcal{S}$ a un système de représentants distincts. (Mendelsohn et Dulmage [1958])

7. Soit G un graphe biparti ayant une bipartition $V(G) = A \dot\cup B$. Soient $S \subseteq A$, $T \subseteq B$, et supposons qu'il y ait un couplage couvrant S et un couplage couvrant T. Montrer qu'il y a un couplage couvrant $S \cup T$.
(Mendelsohn et Dulmage [1958])

8. Montrer qu'un graphe simple avec n sommets et degré minimum k a un couplage de cardinalité $\min\{k, \lfloor \frac{n}{2} \rfloor\}$.
Indication : utiliser le théorème 10.7.

9. Soit G un graphe et M un couplage de G qui n'est pas maximum.

   (a) Montrer qu'il existe $\nu(G) - |M|$ chaînes M-augmentantes dans G.
   Indication : voir la preuve du théorème de Berge 10.7.

   (b) Montrer qu'il existe une chaîne M-augmentante de longueur au plus $\frac{\nu(G)+|M|}{\nu(G)-|M|}$ dans G.

   (c) Soit P une plus courte chaîne M-augmentante dans G, et soit P' une chaîne $(M \triangle E(P))$-augmentante. Alors $|E(P')| \geq |E(P)| + |E(P \cap P')|$.

   Considérons le schéma d'algorithme suivant. Nous partons du couplage vide et à chaque itération nous augmentons la taille du couplage en considérant la plus courte chaîne augmentante. Soient $P_1, P_2, \ldots$ la suite des chaînes augmentantes générées de cette manière. Par (c), $|E(P_k)| \leq |E(P_{k+1})|$ pour tout k.

   (d) Montrer que si $|E(P_i)| = |E(P_j)|$, $i \neq j$, P_i et P_j sont sommet-disjointes.

   (e) Utiliser (b) pour montrer que la suite $|E(P_1)|, |E(P_2)|, \ldots$ contient au plus $2\sqrt{\nu(G)} + 2$ nombres différents.

   (Hopcroft et Karp [1973])

∗ 10. Soit G un graphe biparti ; reportons-nous à l'algorithme de l'exercice 9.

   (a) Montrer que – si M est un couplage – l'union de toutes les plus courtes chaînes M-augmentantes dans G peut se trouver en un temps $O(n + m)$.
   Indication : élaborer une recherche par profondeur d'abord (DFS) avec alternativement des arêtes dans le couplage et non dans le couplage.

   (b) Soit une suite d'itérations de l'algorithme où la longueur de la chaîne augmentante reste constante. Montrer que le temps nécessaire pour effectuer toutes ces itérations n'excède pas $O(n + m)$.
   Indication : appliquer (a) d'abord et trouver ensuite les chaînes successives par DFS. Marquer les sommets déjà visités.

   (c) En combinant (b) et l'exercice 9(e), donner un algorithme en $O\left(\sqrt{n}(m + n)\right)$ pour le PROBLÈME DU COUPLAGE MAXIMUM dans les graphes bipartis.

   (Hopcroft et Karp [1973])

11. Soit G un graphe biparti ayant une bipartition $V(G) = A \dot\cup B$, $A = \{a_1, \ldots, a_k\}$, $B = \{b_1, \ldots, b_k\}$. À tout vecteur $x = (x_e)_{e \in E(G)}$ nous associons une matrice $M_G(x) = (m_{ij}^x)_{1 \le i,j \le k}$ avec

$$m_{ij}^x := \begin{cases} x_e & \text{si } e = \{a_i, b_j\} \in E(G) \\ 0 & \text{sinon} \end{cases}.$$

$\det M_G(x)$ est un polynôme en $x = (x_e)_{e \in E(G)}$. Montrer que G possède un couplage parfait si et seulement si $\det M_G(x)$ n'est pas identiquement zéro.

12. Le **permanent** d'une matrice carrée $M = (m_{ij})_{1 \le i,j \le n}$ est défini par

$$\operatorname{per}(M) := \sum_{\pi \in S_n} \prod_{i=1}^{k} m_{i,\pi(i)},$$

où S_n est l'ensemble des permutations de $\{1, \ldots, n\}$. Montrer qu'un graphe G a $\operatorname{per}(M_G(\mathbb{1}))$ couplages parfaits, $M_G(x)$ étant définie comme précédemment.

13. Une **matrice doublement stochastique** est une matrice carrée non négative dont la somme de chaque colonne et de chaque ligne est égale à 1. Les matrices doublement stochastiques entières sont appelées les **matrices de permutation**.
Falikman [1981] et Egoryčev [1980] ont montré que, si M est une matrice $n \times n$ doublement stochastique,

$$\operatorname{per}(M) \ge \frac{n!}{n^n},$$

et que l'égalité est atteinte quand tous les coefficients de M valent $\frac{1}{n}$. (Cela était une conjecture célèbre de Van der Waerden ; voir aussi Schrijver [1998].)
Brègman [1973] a montré que, si M est une matrice 0-1,

$$\operatorname{per}(M) \le (r_1!)^{\frac{1}{r_1}} \cdot \ldots \cdot (r_n!)^{\frac{1}{r_n}}.$$

(r_i est la somme des éléments de la ligne i). Utiliser ces résultats et l'exercice 12 pour montrer le résultat suivant : Soit G un graphe simple biparti k-régulier ayant $2n$ sommets, et soit $\Phi(G)$ le nombre de couplages parfaits de G. Alors

$$n! \left(\frac{k}{n}\right)^n \le \Phi(G) \le (k!)^{\frac{n}{k}}.$$

14. Montrer que tout graphe 3-régulier avec au plus deux ponts a un couplage parfait. Existe-t-il un graphe 3-régulier sans couplage parfait ?
Indication : utiliser le théorème de Tutte 10.13.
(Petersen [1891])

∗ 15. Soit G un graphe avec $n := |V(G)|$ pair ; montrer que G a un couplage parfait si, pour tout $X \subseteq V(G)$ avec $|X| \le \frac{3}{4}n$,

$$\left| \bigcup_{x \in X} \Gamma(x) \right| \geq \frac{4}{3}|X|.$$

Indication : soit S un ensemble qui viole la condition de Tutte. Montrer que le nombre de composantes connexes ayant un seul élément dans $G - S$ est au plus $\max\left\{0, \frac{4}{3}|S| - \frac{1}{3}n\right\}$. Étudier les cas $|S| \geq \frac{n}{4}$ et $|S| < \frac{n}{4}$ séparément.
(Anderson [1971])

16. Montrer qu'un graphe non orienté G est facteur-critique si et seulement s'il est connexe et $\nu(G) = \nu(G - v)$ pour tout $v \in V(G)$.

17. Montrer que le nombre d'oreilles dans deux décompositions en oreilles impaires de G est le même.

* 18. Soit G un graphe 2-arête-connexe et soit $\varphi(G)$ le nombre minimum d'oreilles paires dans une décomposition en oreilles (voir l'exercice 17(a) du chapitre 2). Montrer que pour tout $e \in E(G)$ $\varphi(G/e) = \varphi(G) + 1$ ou $\varphi(G/e) = \varphi(G) - 1$.
Note : $\varphi(G)$ a été étudiée par Szigeti [1996] et Szegedy et Szegedy [2006].

19. Montrer qu'un graphe facteur critique minimal G (c.-à-d. que le graphe obtenu après suppression de n'importe quelle arête n'est plus facteur-critique) a au plus $\frac{3}{2}(|V(G)| - 1)$ arêtes. Montrer que cette borne est serrée.

20. Montrer comment l'ALGORITHME DE COUPLAGE MAXIMUM D'EDMONDS trouve un couplage dans le graphe de la figure 10.1(b).

21. Peut-on trouver en temps polynomial une couverture par les arêtes de cardinalité minimum dans un graphe ?

* 22. Une arête d'un graphe ayant n sommets sera dite non couplable si elle n'est contenue dans aucun couplage parfait. Comment trouver ces arêtes en $O(n^3)$?
Indication : trouver d'abord un couplage parfait dans G. Puis, pour chaque sommet v, détecter l'ensemble des arêtes non couplables incidentes à v.

23. Soit M un couplage maximum d'un graphe G, et soient F_1, F_2 deux forêts spéciales de blossoms par rapport à M ayant le nombre maximum d'arêtes. Montrer que F_1 et F_2 ont le même nombre de sommets internes.

24. Soit G un graphe k-connexe avec $2\nu(G) < |V(G)| - 1$. Montrer que :
    (a) $\nu(G) \geq k$.
    (b) $\tau(G) \leq 2\nu(G) - k$.
    Indication : utiliser le théorème de Gallai-Edmonds 10.32.
    (Erdős et Gallai [1961])

Références

Littérature générale :

Gerards, A.M.H. [1995] : Matching. In : Handbooks in Operations Research and Management Science ; Volume 7 : Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995, pp. 135–224

Lawler, E.L. [1976] : Combinatorial Optimization ; Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Chapters 5 and 6

Lovász, L., Plummer, M.D. [1986] : Matching Theory. Akadémiai Kiadó, Budapest 1986, and North-Holland, Amsterdam 1986

Papadimitriou, C.H., Steiglitz, K. [1982] : Combinatorial Optimization ; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Chapter 10

Pulleyblank, W.R. [1995] : Matchings and extensions. In : Handbook of Combinatorics ; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam 1995

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 16 and 24

Tarjan, R.E. [1983] : Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Chapter 9

Références citées :

Alt, H., Blum, N., Mehlhorn, K., Paul, M. [1991] : Computing a maximum cardinality matching in a bipartite graph in time $O\left(n^{1.5}\sqrt{m/\log n}\right)$. Information Processing Letters 37 (1991), 237–240

Anderson, I. [1971] : Perfect matchings of a graph. Journal of Combinatorial Theory B 10 (1971), 183–186

Berge, C. [1957] : Two theorems in graph theory. Proceedings of the National Academy of Science of the U.S. 43 (1957), 842–844

Berge, C. [1958] : Sur le couplage maximum d'un graphe. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris) Sér. I Math. 247 (1958), 258–259

Brègman, L.M. [1973] : Certain properties of nonnegative matrices and their permanents. Doklady Akademii Nauk SSSR 211 (1973), 27–30 [in Russian]. English translation : Soviet Mathematics Doklady 14 (1973), 945–949

Dilworth, R.P. [1950] : A decomposition theorem for partially ordered sets. Annals of Mathematics 51 (1950), 161–166

Edmonds, J. [1965] : Paths, trees, and flowers. Canadian Journal of Mathematics 17 (1965), 449–467

Egoryčev, G.P. [1980] : Solution of the Van der Waerden problem for permanents. Soviet Mathematics Doklady 23 (1982), 619–622

Erdős, P., Gallai, T. [1961] : On the minimal number of vertices representing the edges of a graph. Magyar Tudományos Akadémia ; Matematikai Kutató Intézetének Közleményei 6 (1961), 181–203

Falikman, D.I. [1981] : A proof of the Van der Waerden conjecture on the permanent of a doubly stochastic matrix. Matematicheskie Zametki 29 (1981), 931–938 [in Russian]. English translation : Math. Notes of the Acad. Sci. USSR 29 (1981), 475–479

Feder, T., Motwani, R. [1995] : Clique partitions, graph compression and speeding-up algorithms. Journal of Computer and System Sciences 51 (1995), 261–272

Fremuth-Paeger, C., Jungnickel, D. [2003] : Balanced network flows VIII : a revised theory of phase-ordered algorithms and the $O(\sqrt{n}m\log(n^2/m)/\log n)$ bound for the nonbipartite cardinality matching problem. Networks 41 (2003), 137–142

Frobenius, G. [1917] : Über zerlegbare Determinanten. Sitzungsbericht der Königlich Preussischen Akademie der Wissenschaften XVIII (1917), 274–277

Fulkerson, D.R. [1956] : Note on Dilworth's decomposition theorem for partially ordered sets. Proceedings of the AMS 7 (1956), 701–702

Gallai, T. [1959] : Über extreme Punkt- und Kantenmengen. Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae ; Sectio Mathematica 2 (1959), 133–138

Gallai, T. [1964] : Maximale Systeme unabhängiger Kanten. Magyar Tudományos Akadémia ; Matematikai Kutató Intézetének Közleményei 9 (1964), 401–413

Geelen, J.F. [2000] : An algebraic matching algorithm. Combinatorica 20 (2000), 61–70

Geelen, J. Iwata, S. [2005] : Matroid matching via mixed skew-symmetric matrices. Combinatorica 25 (2005), 187–215

Goldberg, A.V., Karzanov, A.V. [2004] : Maximum skew-symmetric flows and matchings. Mathematical Programming A 100 (2004), 537–568

Hall, P. [1935] : On representatives of subsets. Journal of the London Mathematical Society 10 (1935), 26–30

Halmos, P.R., Vaughan, H.E. [1950] : The marriage problem. American Journal of Mathematics 72 (1950), 214–215

Hopcroft, J.E., Karp, R.M. [1973] : An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing 2 (1973), 225–231

König, D. [1916] : Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. Mathematische Annalen 77 (1916), 453–465

König, D. [1931] : Graphs and matrices. Matematikaiés Fizikai Lapok 38 (1931), 116–119 [in Hungarian]

König, D. [1933] : Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen). Acta Litteratum ac Scientiarum Regiae Universitatis Hungaricae Francisco-Josephinae (Szeged). Sectio Scientiarum Mathematicarum 6 (1933), 155–179

Kuhn, H.W. [1955] : The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2 (1955), 83–97

Lovász, L. [1972] : A note on factor-critical graphs. Studia Scientiarum Mathematicarum Hungarica 7 (1972), 279–280

Lovász, L. [1979] : On determinants, matchings and random algorithms. In : Fundamentals of Computation Theory (L. Budach, ed.), Akademie-Verlag, Berlin 1979, pp. 565–574

Mendelsohn, N.S., Dulmage, A.L. [1958] : Some generalizations of the problem of distinct representatives. Canadian Journal of Mathematics 10 (1958), 230–241

Micali, S., Vazirani, V.V. [1980] : An $O(V^{1/2}E)$ algorithm for finding maximum matching in general graphs. Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (1980), 17–27

Mucha, M., Sankowski, P. [2004] : Maximum matchings via Gaussian elimination. Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (2004), 248–255

Mulmuley, K., Vazirani, U.V., Vazirani, V.V. [1987] : Matching is as easy as matrix inversion. Combinatorica 7 (1987), 105–113

Petersen, J. [1891] : Die Theorie der regulären Graphen. Acta Mathematica 15 (1891), 193–220

Rabin, M.O., Vazirani, V.V. [1989] : Maximum matchings in general graphs through randomization. Journal of Algorithms 10 (1989), 557–567

Rizzi, R. [1998] : König's edge coloring theorem without augmenting paths. Journal of Graph Theory 29 (1998), 87

Schrijver, A. [1998] : Counting 1-factors in regular bipartite graphs. Journal of Combinatorial Theory B 72 (1998), 122–135

Sperner, E. [1928] : Ein Satz über Untermengen einer endlichen Menge. Mathematische Zeitschrift 27 (1928), 544–548

Szegedy, B., Szegedy, C. [2006] : Symplectic spaces and ear-decomposition of matroids. Combinatorica 26 (2006), 353–377

Szigeti, Z. [1996] : On a matroid defined by ear-decompositions. Combinatorica 16 (1996), 233–241

Tutte, W.T. [1947] : The factorization of linear graphs. Journal of the London Mathematical Society 22 (1947), 107–111

Vazirani, V.V. [1994] : A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. Combinatorica 14 (1994), 71–109

Chapitre 11

Couplage avec poids

Dans la classe des problèmes d'optimisation combinatoire résolubles en temps polynomial, le couplage avec poids est un des plus «difficiles». Nous allons étendre l'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS au cas pondéré et obtenir encore une implémentation en $O(n^3)$. Cet algorithme a de nombreuses applications ; certaines d'entre elles sont mentionnées dans les exercices et au chapitre 12.2. Il existe deux formulations classiques pour ce problème :

PROBLÈME DU COUPLAGE DE POIDS MAXIMUM	
Instance	Un graphe non orienté G et des poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver un couplage de poids maximum dans G.

PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM	
Instance	Un graphe non orienté G et des poids $c : E(G) \to \mathbb{R}$.
Tâche	Trouver un couplage parfait de poids minimum G ou conclure que G n'a pas de couplage parfait.

Il est facile de voir que ces deux problèmes sont équivalents : Soit (G, c) une instance du PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM ; posons $c'(e) := K - c(e)$ pour tout $e \in E(G)$, avec $K := 1 + \sum_{e \in E(G)} |c(e)|$. Tout couplage de poids maximum dans (G, c') est aussi de cardinalité maximum, et fournit une solution au PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM (G, c).

Inversement, soit (G, c) une instance du PROBLÈME DU COUPLAGE DE POIDS MAXIMUM. Additionnons $|V(G)|$ nouveaux sommets et toutes les arêtes manquantes afin d'obtenir un graphe complet G' avec $2|V(G)|$ sommets. Posons $c'(e) := -c(e)$ pour tout $e \in E(G)$ et $c'(e) := 0$ pour toutes les nouvelles arêtes e. Le couplage obtenu en supprimant les arêtes d'un couplage parfait de poids minimum de (G', c') qui ne sont pas dans G est un couplage de poids maximum dans (G, c).

Nous n'étudierons que le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM. Comme dans le chapitre précédent, nous commencerons, au paragraphe 11.1, par les graphes bipartis. Après un aperçu de l'algorithme de couplage de poids maximum présenté au paragraphe 11.2, nous donnerons, dans le paragraphe 11.3, des détails sur l'implémentation nécessaires pour obtenir une complexité $O(n^3)$. Quelquefois, on souhaite résoudre plusieurs problèmes de couplage qui diffèrent seulement sur quelques arcs ; il n'est pas alors nécessaire de reprendre à zéro la résolution de chaque problème, comme nous le verrons paragraphe 11.4. Enfin, au paragraphe 11.5 nous étudierons le polytope du couplage, c.-à-d. l'enveloppe convexe des vecteurs d'incidence des couplages. L'algorithme de couplage de poids maximum s'appuie implicitement sur la description de ce polytope et inversement la description du polytope est une conséquence directe de l'algorithme.

11.1 Problème d'affectation

Le PROBLÈME D'AFFECTATION est une autre dénomination du PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM dans les graphes bipartis. C'est un problème classique de l'optimisation combinatoire ; son histoire remonte probablement aux travaux de Monge [1784].

Comme dans la preuve du théorème 10.5, le problème d'affectation se ramène à un problème de flot :

Théorème 11.1. *Le* PROBLÈME D'AFFECTATION *peut être résolu en un temps* $O(nm + n^2 \log n)$.

Preuve. Soit G un graphe biparti ayant une bipartition $V(G) = A \dot\cup B$. Nous supposons que $|A| = |B| = n$. Ajoutons un sommet s que nous connectons à tous les sommets de A, ajoutons un sommet t que nous connectons à tous les sommets de B. Orientons les arêtes de s vers A, de A vers B, et de B vers t. Mettons partout des capacités égales à 1, et affectons un coût nul aux nouvelles arêtes.

Alors tout flot entier de s à t de valeur n correspond à un couplage parfait de même coût, et inversement. Nous avons donc à résoudre le PROBLÈME DU FLOT DE COÛT MINIMUM. Pour cela nous appliquons l'ALGORITHME PAR PLUS COURTS CHEMINS SUCCESSIFS (voir paragraphe 9.4). La demande totale est n. Donc par le théorème 9.12, le temps de calcul est $O(nm + n^2 \log n)$. □

Cet algorithme est le plus rapide des algorithmes connus. Il est équivalent à la «méthode hongroise» de Kuhn [1955] et Munkres [1957] qui est le plus ancien algorithme polynomial pour le PROBLÈME D'AFFECTATION (voir exercice 9).

Il est intéressant d'étudier le PROBLÈME D'AFFECTATION par le biais de la PROGRAMMATION LINÉAIRE. En effet, dans la formulation comme PL en nombres entiers

$$\min \left\{ \sum_{e \in E(G)} c(e) x_e : x_e \in \{0,1\} \ (e \in E(G)), \sum_{e \in \delta(v)} x_e = 1 \ (v \in V(G)) \right\},$$

les contraintes d'intégrité peuvent être omises (remplacer $x_e \in \{0,1\}$ par $x_e \geq 0$) :

Théorème 11.2. *Soit G un graphe, et soient*

$$P := \left\{ x \in \mathbb{R}_+^{E(G)} : \sum_{e \in \delta(v)} x_e \leq 1 \text{ pour tout } v \in V(G) \right\} \text{ et}$$

$$Q := \left\{ x \in \mathbb{R}_+^{E(G)} : \sum_{e \in \delta(v)} x_e = 1 \text{ pour tout } v \in V(G) \right\}$$

le **polytope fractionnaire du couplage** *et le* **polytope fractionnaire des couplages parfaits** *de G. Si G est biparti, alors P et Q sont des polytopes entiers.*

Preuve. Si G est biparti, sa matrice d'incidence M est totalement unimodulaire par le théorème 5.25. Donc P est entier par le théorème de Hoffman-Kruskal 5.20. Q qui est une face de P est également entier. □

Il existe une application élégante de ce théorème aux matrices doublement stochastiques. Une **matrice doublement stochastique** est une matrice carrée non négative dont la somme des éléments de chaque ligne et de chaque colonne vaut 1. Les matrices doublement stochastiques entières sont les **matrices de permutation**.

Corollaire 11.3. (Birkhoff [1946], von Neumann [1953]) *Toute matrice doublement stochastique M peut s'écrire comme combinaison convexe de matrices de permutation $P_1, \ldots, P_k$ (c.-à-d. $M = c_1P_1 + \ldots + c_kP_k$ où $c_1, \ldots, c_k$ sont des nombres non négatifs tels que $c_1 + \ldots + c_k = 1$).*

Preuve. Soit $M = (m_{ij})_{i,j \in \{1,\ldots,n\}}$ une matrice $n \times n$ doublement stochastique ; soit $K_{n,n}$ le graphe biparti complet ayant comme bipartition $\{a_1, \ldots, a_n\} \dot\cup \{b_1, \ldots, b_n\}$. Associons à $e = (a_i, b_j) \in E(K_{n,n})$ la variable $x_e = m_{ij}$. Puisque M est doublement stochastique, x est dans le polytope fractionnaire des couplages parfaits Q de $K_{n,n}$. Par le théorème 11.2 et le corollaire 3.32, x est combinaison convexe de sommets entiers de Q correspondant aux matrices de permutation. □

Ce corollaire peut se démontrer directement (exercice 3).

11.2 Aperçu de l'algorithme du couplage avec poids

Le but de ce paragraphe et du suivant est de présenter un algorithme polynomial pour le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM, trouvé par Edmonds [1965] ; cet algorithme utilise des concepts développés dans l'algorithme du PROBLÈME DU COUPLAGE MAXIMUM (section 10.5).

Donnons d'abord les idées générales en laissant de côté les questions d'implémentation. Soit G un graphe muni de poids $c : E(G) \to \mathbb{R}$; le PROBLÈME DU

COUPLAGE PARFAIT DE POIDS MINIMUM peut se formuler comme PL en nombres entiers

$$\min \left\{ \sum_{e \in E(G)} c(e) x_e : x_e \in \{0,1\} \ (e \in E(G)), \ \sum_{e \in \delta(v)} x_e = 1 \ (v \in V(G)) \right\}.$$

Si A est un sous-ensemble de $V(G)$ de cardinalité impaire, tout couplage parfait a un nombre impair d'arêtes dans $\delta(A)$, en particulier au moins une. Donc la contrainte

$$\sum_{e \in \delta(A)} x_e \ \geq \ 1$$

peut être rajoutée. Nous utiliserons dans ce chapitre la notation $\mathcal{A} := \{A \subseteq V(G) : |A| \text{ impair}\}$. Considérons la relaxation par PL :

$$\begin{array}{lll} \min & \displaystyle\sum_{e \in E(G)} c(e) x_e & \\ \text{s.c.} & x_e \ \geq \ 0 & (e \in E(G)) \\ & \displaystyle\sum_{e \in \delta(v)} x_e \ = \ 1 & (v \in V(G)) \\ & \displaystyle\sum_{e \in \delta(A)} x_e \ \geq \ 1 & (A \in \mathcal{A}, \ |A| > 1). \end{array} \tag{11.1}$$

Nous montrerons plus loin que le polytope décrit par (11.1) est entier ; ce PL décrit donc le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM (ce sera le théorème 11.13, un résultat important de ce chapitre). Dans ce qui suit, nous n'aurons pas besoin de ce résultat, mais nous raisonnerons néanmoins sur ce PL.

Pour construire le dual de (11.1), introduisons une variable z_A associée à chaque contrainte du dual induite par chaque $A \in \mathcal{A}$. Le dual s'écrit alors :

$$\begin{array}{lll} \max & \displaystyle\sum_{A \in \mathcal{A}} z_A & \\ \text{s.c.} & z_A \ \geq \ 0 & (A \in \mathcal{A}, \ |A| > 1) \\ & \displaystyle\sum_{A \in \mathcal{A} : e \in \delta(A)} z_A \ \leq \ c(e) & (e \in E(G)). \end{array} \tag{11.2}$$

Notons que les variables duales $z_{\{v\}}$ pour $v \in V(G)$ ne sont pas astreintes à être non négatives. L'algorithme d'Edmonds est un algorithme primal-dual. Le couplage initial est : $x_e = 0$ pour tout $e \in E(G)$ et la solution initiale du dual est

$$z_A \ := \ \begin{cases} \frac{1}{2} \min\{c(e) : e \in \delta(A)\} & \text{si } |A| = 1 \\ \\ 0 & \text{sinon} \end{cases} .$$

À chaque étape de l'algorithme, z sera une solution duale réalisable et

$$\begin{aligned} x_e > 0 \quad &\Rightarrow \quad \sum_{A\in\mathcal{A}:e\in\delta(A)} z_A = c(e); \\ z_A > 0 \quad &\Rightarrow \quad \sum_{e\in\delta(A)} x_e \leq 1. \end{aligned} \tag{11.3}$$

L'algorithme s'arrêtera quand x sera le vecteur d'incidence d'un couplage parfait (nous aurons alors la réalisabilité pour le primal). Par la condition des écarts complémentaires (11.3) (corollaire 3.23), nous aurons l'optimalité pour le primal et le dual. Comme x est optimal pour (11.1) et aussi entier, x est le vecteur d'incidence d'un couplage parfait de poids minimum.

Si z est une solution duale réalisable, nous dirons que e est **serrée** si la contrainte duale associée à e est satisfaite à égalité, c.-à-d. si

$$\sum_{A\in\mathcal{A}:e\in\delta(A)} z_A = c(e).$$

À chaque étape la solution courante sera un couplage dont les arêtes sont serrées.

Nous associerons à z un graphe G_z résultant de G en supprimant toutes les arêtes non serrées et en contractant chaque ensemble B avec $z_B > 0$ en un sommet. La famille $\mathcal{B} := \{B \in \mathcal{A} : z_B > 0\}$ sera laminaire, et chaque élément de $\mathcal{B}$ induira un sous-graphe facteur-critique constitué uniquement d'arêtes critiques. Au début $\mathcal{B}$ est constituée des singletons.

Chaque itération se déroulera de la manière suivante : nous chercherons d'abord un couplage de cardinalité maximum M dans G_z, en utilisant l'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS. Si M est un couplage parfait, nous pourrons ajouter des arêtes serrées à M afin d'obtenir un couplage parfait dans G. Puisque les conditions (11.3) seront vérifiées, ce couplage sera optimal.

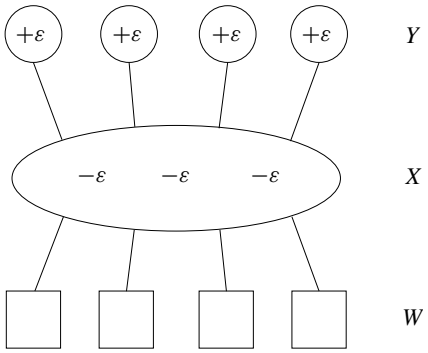


Figure 11.1.

Sinon nous étudierons la décomposition de Gallai-Edmonds W, X, Y de G_z (théorème 10.32). Pour chaque sommet v de G_z soit $B(v) \in \mathcal{B}$ l'ensemble des

sommets de G qui a été contracté en v. Nous modifierons comme suit la solution duale (voir l'illustration sur la figure 11.1) : pour tout $v \in X$ nous retrancherons à $z_{B(v)}$ une constante positive ε, pour toute composante connexe C de $G_z[Y]$ nous ajouterons ε à z_A où $A = \bigcup_{v \in C} B(v)$.

Les arêtes serrées resteront serrées, puisque toutes les arêtes serrées avec une extrémité dans X ont leur autre extrémité dans Y par le théorème 10.32. (Toutes les arêtes de la forêt alternée sur laquelle nous travaillerons resteront serrées.)

Nous choisirons pour ε la plus grande valeur possible afin de préserver la réalisabilité du dual. Puisque le graphe à l'itération actuelle n'est pas un couplage parfait, le nombre de composantes connexes $G_z[Y]$ sera supérieur à $|X|$. Donc la modification dans le dual accroîtra la valeur de la fonction objectif du dual $\sum_{A \in \mathcal{A}} z_A$ par au moins ε. Si ε peut être choisi arbitrairement grand, le dual (11.2) sera non borné, le primal (11.1) ne sera pas réalisable (théorème 3.27) et G n'aura pas de couplage parfait.

Comme la solution duale sera modifiée, le graphe G_z sera aussi modifié : de nouvelles arêtes deviendront serrées, de nouveaux sommets seront contractés (associés aux composantes Y qui ne sont pas des singletons), et certains ensembles contractés pourront être «défaits» (les non-singletons de X dont les variables duales deviennent égales à zéro).

Ce processus se poursuivra jusqu'à l'obtention d'un couplage parfait. Nous montrerons que cette procédure est finie. Cela se déduira du fait que, entre deux augmentations, chaque étape (grossir, contracter, défaire) augmente le nombre de sommets externes.

11.3 Implémentation de l'algorithme du couplage avec poids

Venons-en maintenant aux détails de l'implémentation. Comme pour L'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS nous ne contracterons pas explicitement les blossoms, mais nous garderons trace de ceux-ci à travers leurs décompositions en oreilles. Il y a cependant de nombreuses difficultés.

L'opération «contracter» de l'ALGORITHME DU COUPLAGE MAXIMUM D'EDMONDS produit un blossom externe. Après l'étape «augmenter», deux composantes connexes de la forêt de blossoms sont placées hors de la forêt. Comme la solution duale ne change pas, ces blossoms resteront des blossoms que nous appellerons blossoms hors de la forêt. L'opération «grossir» peut concerner des blossoms hors de la forêt qui deviendront alors des blossoms internes ou externes. Nous devrons donc travailler sur des forêts générales de blossoms.

Un autre problème est que nous aurons à défaire certains blossoms : en effet, si A est un blossom interne tel que z_A devienne égal à zéro, il se peut qu'il y ait des sous-ensembles $A' \subseteq A$ avec $|A'| > 1$ et $z_{A'} > 0$. Il nous faudra alors défaire A, sans toucher aux plus petits blossoms qui sont dans A (sauf s'ils restent internes et si leurs variables duales valent zéro).

Durant l'algorithme, nous aurons une famille laminaire $\mathcal{B} \subseteq \mathcal{A}$, contenant au moins tous les singletons. Les éléments de $\mathcal{B}$ sont des blossoms. $z_A = 0$ si $A \notin \mathcal{B}$.

L'ensemble laminaire $\mathcal{B}$ est décrit par une représentation par arbre (voir proposition 2.14). Un nombre sera affecté à chaque blossom $\mathcal{B}$ qui n'est pas un singleton.

Nous mémoriserons à chaque étape de l'algorithme les décompositions en oreilles des blossoms de $\mathcal{B}$. Les variables $\mu(x)$ pour $x \in V(G)$ définissent, comme dans le chapitre précédent le couplage courant M. Nous noterons par $b^1(x), \ldots, b^{k_x}(x)$ les blossoms de $\mathcal{B}$ contenant x, sans le singleton, $b^{k_x}(x)$ étant le plus grand de ces blossoms. Nous avons des variables $\rho^i(x)$ et $\varphi^i(x)$ pour chaque $x \in V(G)$ et $i = 1, \ldots, k_x$. $\rho^i(x)$ est la base du blossom $b^i(x)$. $\mu(x)$ et $\varphi^j(x)$, pour tous les x et j avec $b^j(x) = i$, sont associés à une décomposition en oreilles M-alternée du blossom i.

Nous devons mettre à jour les structures des blossoms (φ et ρ) après chaque augmentation. Actualiser ρ est simple. Actualiser φ peut également se faire en temps linéaire par le lemme 10.23.

Pour les blossoms internes, nous avons besoin, en plus de la base, du sommet le plus proche de la racine de l'arbre dans la forêt générale de blossoms et du voisin de ce dernier sommet dans le blossom externe suivant. Ces sommets seront notés $\sigma(x)$ et $\chi(\sigma(x))$ pour chaque base x d'un blossom interne ; voir la figure 11.2 comme illustration.

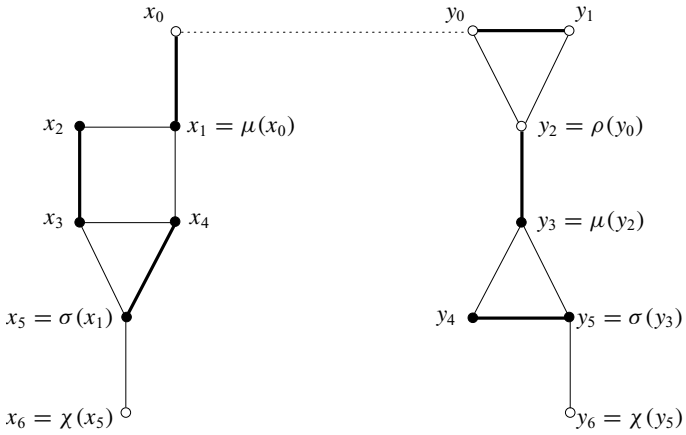


Figure 11.2.

Grâce à ces variables, les chaînes alternées vers la racine de l'arbre peuvent être détectées. Puisque les blossoms doivent rester des blossoms après une augmentation, il nous faudra choisir la chaîne augmentante qui garantit que chaque blossom continuera à avoir un couplage presque parfait.

La figure 11.2 montre qu'il faut prendre des précautions : il y a deux blossoms internes, induits par $\{x_3, x_4, x_5\}$ et $\{x_1, x_2, x_3, x_4, x_5\}$. Si nous considérons seulement la décomposition en oreilles du blossom extérieur pour trouver une chaîne alternée de x_0 vers la racine x_6, nous obtiendrons $(x_0, x_1, x_4, x_5 = \sigma(x_1), x_6 =$

$\chi(x_5)$). Après augmentation le long de $(y_6, y_5, y_4, y_3, y_2, y_1, y_0, x_0, x_1, x_4, x_5, x_6)$, le sous-graphe induit par $\{x_3, x_4, x_5\}$ ne contient plus de couplage presque parfait.

Il faut donc trouver à l'intérieur de chaque blossom une chaîne alternée qui contient un nombre pair d'arêtes dans chaque sous-blossom. Cela est l'objet de la procédure suivante :

BLOSSOMPATH

Input Un sommet x_0.

Output Une chaîne M-alternée $Q(x_0)$ de x_0 à $\rho^{k_{x_0}}(x_0)$.

① $h := 0$ et $B := \{b^j(x_0) : j = 1, \ldots, k_{x_0}\}$.

② **While** $x_{2h} \neq \rho^{k_{x_0}}(x_0)$ **do** :
$x_{2h+1} := \mu(x_{2h})$ et $x_{2h+2} := \varphi^i(x_{2h+1})$, où
$i = \min\{j \in \{1, \ldots, k_{x_{2h+1}}\} : b^j(x_{2h+1}) \in B\}$.
Ajouter à B tous les blossoms de $\mathcal{B}$ qui contiennent x_{2h+2} mais pas x_{2h+1}.
Supprimer de B les blossoms dont la base est x_{2h+2}.
$h := h + 1$.

③ $Q(x_0)$ est la chaîne de sommets $x_0, x_1, \ldots, x_{2h}$.

Proposition 11.4. *La procédure* BLOSSOMPATH *peut être implémentée en un temps $O(n)$. $M \triangle E(Q(x_0))$ contient un couplage presque parfait dans chaque blossom.*

Preuve. Vérifions d'abord que la procédure calcule bien une chaîne : quand la chaîne sort d'un blossom $\mathcal{B}$, elle n'y entrera plus jamais parce que, quand on contracte les sous-blossoms maximaux d'un blossom dans $\mathcal{B}$, on obtient un cycle (une propriété qui sera toujours vraie).

Au début de chaque itération, B est la liste de tous les blossoms qui contiennent x_0 ou qui sont ceux dans lesquels on est entré par une arête qui n'est pas dans le couplage sans en être ressorti. La chaîne construite sort de tout blossom dans B par une arête du couplage. Donc le nombre d'arêtes à l'intérieur de chaque blossom est pair, ce qui démontre la deuxième affirmation de la proposition.

Le seul problème pour la complexité est la mise à jour de B. B est stocké suivant une liste triée. Utilisant la représentation par arbre de $\mathcal{B}$ et le fait qu'on entre et sort de chaque blossom au plus une fois, le temps de calcul est en $O(n + |\mathcal{B}|)$. Comme $\mathcal{B}$ est laminaire, $|\mathcal{B}| = O(n)$; la complexité est donc linéaire. □

Pour trouver une chaîne augmentante, il suffit d'appliquer la procédure BLOSSOMPATH à l'intérieur des blossoms, et d'utiliser μ et χ entre les blossoms. Quand nous trouvons d'abord des sommets externes x, y adjacents dans des arbres distincts de la forêt générale de blossoms, nous appliquons la procédure précédente à x et à y. L'union des deux chaînes et de l'arête (x, y) sera la chaîne augmentante.

TREEPATH	
Input	Un sommet externe v.
Output	Une chaîne alternée $P(v)$ de v à la racine dans la forêt de blossoms.

① Initialement $P(v)$ est le sommet v. Soit $x := v$.

② $y := \rho^{k_x}(x)$. $Q(x) :=$ BLOSSOMPATH(x). Ajouter $Q(x)$ à $P(v)$.
If $\mu(y) = y$ **then stop**.

③ $P(v) := P(v) + (y, \mu(y))$.
$Q(\sigma(\mu(y))) :=$ BLOSSOMPATH$(\sigma(\mu(y)))$.
Ajouter à $P(v)$ la chaîne $Q(\sigma(\mu(y)))$ décrite dans le sens inverse.
$P(v) := P(v) + (\sigma(\mu(y)), \chi(\sigma(\mu(y))))$.
$x := \chi(\sigma(\mu(y)))$ et **go to** ②.

Le deuxième problème consiste à trouver efficacement ε. La forêt de blossoms, une fois toutes les opérations grossir, contracter, augmenter effectuées, permet d'obtenir la décomposition de Gallai-Edmonds W, X, Y de G_z. W contient les blossoms hors de la forêt, X les blossoms internes, et Y les blossoms externes.

Pour simplifier nos notations, posons $c((v,w)) := \infty$ si $(v,w) \notin E(G)$. De plus, nous utiliserons l'abréviation

$$\acute{e}cart(v,w) \;:=\; c((v,w)) - \sum_{A\in\mathcal{A},\,(v,w)\in\delta(A)} z_A.$$

Ainsi (v,w) est une arête serrée si et seulement si $\acute{e}cart(v,w) = 0$. Soit alors :

$$\begin{aligned}
\varepsilon_1 &:= \min\{z_A : A \text{ est un blossom interne maximal, } |A| > 1\};\\
\varepsilon_2 &:= \min\{\acute{e}cart(x,y) : x \text{ externe, } y \text{ hors de la forêt}\};\\
\varepsilon_3 &:= \frac{1}{2}\min\{\acute{e}cart(x,y) : x, y \text{ externes, dans des blossoms différents}\};\\
\varepsilon &:= \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}.
\end{aligned}$$

ε est le nombre maximum tel que la modification du dual par ε maintient la réalisabilité du dual. Si $\varepsilon = \infty$, (11.2) est non borné (11.1) est non réalisable. Dans ce cas, G n'a pas de couplage parfait.

ε peut être calculé en temps fini. Cependant, si nous souhaitons un temps de calcul total $O(n^3)$, il nous faudra calculer ε en un temps $O(n)$. Cela est facile pour ε_1, mais pour ε_2 et ε_3 il nous faudra utiliser des structures de données additionnelles. Si $A \in \mathcal{B}$ soit

$$\zeta_A \;:=\; \sum_{B\in\mathcal{B}:A\subseteq B} z_B.$$

Nous actualiserons ces valeurs à chaque modification du dual ; cela se fera aisément en temps linéaire (en utilisant la représentation par arbre de $\mathcal{B}$). On a alors

$$\varepsilon_2 \;=\; \min\big\{c((x,y)) - \zeta_{\{x\}} - \zeta_{\{y\}} : x \text{ externe, } y \text{ hors de la forêt}\big\}$$

$$\varepsilon_3 \;=\; \frac{1}{2}\min\big\{c((x,y)) - \zeta_{\{x\}} - \zeta_{\{y\}} : x, y \text{ externes, dans différents blossoms}\big\}.$$

Pour calculer ε_2, nous stockons, pour chaque sommet v hors de la forêt, le voisin externe w pour lequel $\acute{e}cart(v,w) = c((v,w)) - \zeta_{\{v\}} - \zeta_{\{w\}}$ est minimum. Nous appellerons τ_v ce voisin. Ces variables seront mises à jour si nécessaire. Il est alors facile de calculer $\varepsilon_2 = \min\{c((v,\tau_v)) - \zeta_{\{v\}} - \zeta_{\{\tau_v\}} : v \text{ hors de la forêt}\}$.

Pour calculer ε_3, nous introduisons les variables t_v^A et τ_v^A pour chaque sommet externe v et pour chaque $A \in \mathcal{B}$, à moins que A soit externe mais non maximal. τ_v^A est le sommet en A qui minimise $\acute{e}cart(v,\tau_v^A)$, et $t_v^A = \acute{e}cart(v,\tau_v^A) + \Delta + \zeta_A$, où Δ est la somme de tous les ε dans les modifications précédentes du dual.

Bien que le calcul de ε_3 ne dépende que des variables t_v^A associées aux blossoms externes maximaux de $\mathcal{B}$, nous mettons à jour également ces variables pour les blossoms internes et hors de la forêt (même ceux qui ne sont pas maximaux), parce qu'ils peuvent devenir externes et maximaux plus tard. Les blossoms externes non maximaux ne peuvent devenir maximaux avant une augmentation. Après chaque augmentation, il faudra cependant recalculer toutes ces variables.

La variable t_v^A a la valeur $\acute{e}cart(v,\tau_v^A) + \Delta + \zeta_A$ à chaque instant. Observons que cette valeur ne change pas tant que v reste externe, $A \in \mathcal{B}$, et τ_v^A est le sommet dans A minimisant $\acute{e}cart(v,\tau_v^A)$. Finalement nous avons $t^A := \min\{t_v^A : v \notin A,\ v \text{ externe}\}$. En conclusion

$$\varepsilon_3 \;=\; \frac{1}{2}\acute{e}cart(v,\tau_v^A) \;=\; \frac{1}{2}(t_v^A - \Delta - \zeta_A) \;=\; \frac{1}{2}(t^A - \Delta - \zeta_A),$$

où A est un élément externe maximal de $\mathcal{B}$ tel que $t^A - \zeta_A$ soit minimum et tel que v soit un sommet externe avec $v \notin A$ et $t_v^A = t^A$.

À certaines étapes nous devrons actualiser τ_v^A et t_v^A pour un certain sommet v et tous les $A \in \mathcal{B}$ (sauf ceux qui sont externes et non maximaux), par exemple si un nouveau sommet devient externe. La procédure suivante met également à jour, si nécessaire, les variables τ_w pour les sommets hors de la forêt w.

UPDATE

Input Un sommet externe v.

Output Mise à jour de τ_v^A, t_v^A et t^A pour tout $A \in \mathcal{B}$ et τ_w pour tous les sommets hors de la forêt w.

① **For** chaque voisin w de v qui est hors de la forêt **do** :
If $c((v,w)) - \zeta_{\{v\}} < c((w,\tau_w)) - \zeta_{\{\tau_w\}}$ **then** $\tau_w := v$.

② **For** tout $x \in V(G)$ **do** : $\tau_v^{\{x\}} := x$ et $t_v^{\{x\}} := c((v,x)) - \zeta_{\{v\}} + \Delta$.

③ **For** $A \in \mathcal{B}$ avec $|A| > 1$ **do** :
Calculer inductivement $\tau_v^A := \tau_v^{A'}$ et $t_v^A := t_v^{A'} - \zeta_{A'} + \zeta_A$, où A' est l'un des sous-ensembles maximaux de A dans $\mathcal{B}$ pour lequel $t_v^{A'} - \zeta_{A'}$ est minimum.

④ **For** $A \in \mathcal{B}$ avec $v \notin A$, à l'exception de ceux qui sont externes et non maximaux, **do** : $t^A := \min\{t^A, t_v^A\}$.

Cette procédure met clairement à jour τ_v^A et t_v^A. Il est important qu'elle s'exécute en temps linéaire :

Lemme 11.5. *Si $\mathcal{B}$ est laminaire, la procédure* UPDATE *est implémentable en un temps $O(n)$.*

Preuve. Par la proposition 2.15, une famille laminaire de sous-ensembles de $V(G)$ a une taille $2|V(G)| = O(n)$. Si $\mathcal{B}$ est représenté par une représentation par arbre, on obtient facilement une implémentation en temps linéaire. □

Nous pouvons maintenant décrire complètement l'algorithme. Au lieu d'identifier les sommets internes et externes à travers les valeurs de μ, ϕ et ρ, nous attribuerons à chaque sommet son statut (externe, interne, hors de la forêt).

ALGORITHME DU COUPLAGE PARFAIT DE POIDS MINIMUM	
Input	Un graphe G, des poids $c : E(G) \to \mathbb{R}$.
Output	Un couplage parfait de poids minimum dans G, donné par ses arêtes $(x, \mu(x))$, ou la réponse que G n'a pas de couplage parfait.

① $\mathcal{B} := \{\{v\} : v \in V(G)\}$ et $K := 0$. $\Delta := 0$.
$z_{\{v\}} := \frac{1}{2}\min\{c(e) : e \in \delta(v)\}$ et $\zeta_{\{v\}} := z_{\{v\}}$ pour tout $v \in V(G)$.
$k_v := 0$, $\mu(v) := v$, $\rho^0(v) := v$, et $\varphi^0(v) := v$ pour tout $v \in V(G)$.
Marquer tous les sommets comme externes.

② **For** tout $v \in V(G)$ **do** : $balayage(v) :=$ *faux*.
For tout sommet hors de la forêt v **do** : Soit τ_v un sommet externe arbitraire.
$t^A := \infty$ pour tout $A \in \mathcal{B}$.
For tous les sommets externes v **do** : UPDATE(v).

③ **If** tous les sommets sont balayés
then go to ⑧,
else soit x un sommet externe avec $balayage(x) =$ *faux*.

④ Soit y un voisin de x tel que (x, y) soit serré et soit y est hors de la forêt ou (y est externe et $\rho^{k_y}(y) \neq \rho^{k_x}(x)$).
If un tel y n'existe pas **then** $balayage(x) :=$ *vrai* et **go to** ③.

⑤ **If** y n'est pas hors de la forêt **then go to** ⑥, **else** :
(«grossir»)
$\sigma(\rho^{k_y}(y)) := y$ et $\chi(y) := x$.
Marquer tous les sommets v avec $\rho^{k_v}(v) = \rho^{k_y}(y)$ comme internes.
Marquer tous les sommets v avec $\mu(\rho^{k_v}(v)) = \rho^{k_y}(y)$ comme externes.
For tout nouveau sommet externe v **do** : UPDATE(v).
Go to ④.

⑥ Soit $P(x) :=$ TREEPATH(x) donné par $(x = x_0, x_1, x_2, \ldots, x_{2h})$.
Soit $P(y) :=$ TREEPATH(y) donné par $(y = y_0, y_1, y_2, \ldots, y_{2j})$.
If $P(x)$ et $P(y)$ ne sont pas sommet-disjoints **then go to** ⑦, **else** :
(«augmenter»)
For $i := 0$ **to** $h-1$ **do** : $\mu(x_{2i+1}) := x_{2i+2}$ et $\mu(x_{2i+2}) := x_{2i+1}$.
For $i := 0$ **to** $j-1$ **do** : $\mu(y_{2i+1}) := y_{2i+2}$ et $\mu(y_{2i+2}) := y_{2i+1}$.
$\mu(x) := y$ et $\mu(y) := x$.
Marquer comme hors de la forêt tous les sommets v tels que l'extrémité de TREEPATH(v) est x_{2h} ou y_{2j}. Actualiser toutes les valeurs $\varphi^i(v)$ et $\rho^i(v)$ pour ces sommets (utilisant le lemme 10.23).
If $\mu(v) \neq v$ pour tout v **then stop**, **else go to** ②.

⑦ («contracter»)
Soit $r = x_{2h'} = y_{2j'}$ le premier sommet externe de $V(P(x)) \cap V(P(y))$ avec $\rho^{k_r}(r) = r$.
Soit $A := \{v \in V(G) : \rho^{k_v}(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})\}$.
$K := K+1$, $\mathcal{B} := \mathcal{B} \cup \{A\}$, $z_A := 0$ et $\zeta_A := 0$.
For tout $v \in A$ **do** :
$k_v := k_v + 1$, $b^{k_v}(v) := K$, $\rho^{k_v}(v) := r$, $\varphi^{k_v}(v) := \varphi^{k_v - 1}(v)$ et marquer v comme externe.
For $i := 1$ **to** h' **do** :
If $\rho^{k_{x_{2i}}-1}(x_{2i}) \neq r$ **then** $\varphi^{k_{x_{2i}}}(x_{2i}) := x_{2i-1}$.
If $\rho^{k_{x_{2i-1}}-1}(x_{2i-1}) \neq r$ **then** $\varphi^{k_{x_{2i-1}}}(x_{2i-1}) := x_{2i}$.
For $i := 1$ **to** j' **do** :
If $\rho^{k_{y_{2i}}-1}(y_{2i}) \neq r$ **then** $\varphi^{k_{y_{2i}}}(y_{2i}) := y_{2i-1}$.
If $\rho^{k_{y_{2i-1}}-1}(y_{2i-1}) \neq r$ **then** $\varphi^{k_{y_{2i-1}}}(y_{2i-1}) := y_{2i}$.
If $\rho^{k_x-1}(x) \neq r$ **then** $\varphi^{k_x}(x) := y$.
If $\rho^{k_y-1}(y) \neq r$ **then** $\varphi^{k_y}(y) := y$.
For chaque sommet externe v **do** :
$t_v^A := t_v^{A'} - \zeta_{A'}$ et $\tau_v^A := \tau_v^{A'}$, où A' est le sous-ensemble propre maximal de A dans $\mathcal{B}$ tel que $t_v^{A'} - \zeta_{A'}$ soit minimum.
$t^A := \min\{t_v^A : v$ externe, il n'existe aucun $\bar{A} \in \mathcal{B}$ tel que $A \cup \{v\} \subseteq \bar{A}\}$.
For chaque nouveau sommet externe v **do** : UPDATE(v).
Go to ④.

⑧ («changement du dual»)
$\varepsilon_1 := \min\{z_A : A$ élément interne maximal de $\mathcal{B}$, $|A| > 1\}$.
$\varepsilon_2 := \min\{c((v,\tau_v)) - \zeta_{\{v\}} - \zeta_{\{\tau_v\}} : v$ hors de la forêt$\}$.
$\varepsilon_3 := \min\{\frac{1}{2}(t^A - \Delta - \zeta_A) : A$ élément externe maximal de $\mathcal{B}\}$.
$\varepsilon := \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$. **If** $\varepsilon = \infty$, **then stop** (G n'a pas de couplage parfait).
If $\varepsilon = \varepsilon_2 = c((v,\tau_v)) - \zeta_{\{v\}} - \zeta_{\{\tau_v\}}$, v externe **then** $balayage(\tau_v) :=$ *faux*.
If $\varepsilon = \varepsilon_3 = \frac{1}{2}(t_v^A - \Delta - \zeta_A)$, A élément externe maximal de $\mathcal{B}$, v externe et $v \notin A$ **then** $balayage(v) :=$ *faux*.
For chaque élément externe maximal de A de $\mathcal{B}$ **do** :
$z_A := z_A + \varepsilon$ et $\zeta_{A'} := \zeta_{A'} + \varepsilon$ pour tout $A' \in \mathcal{B}$ avec $A' \subseteq A$.
For chaque élément interne maximal de A de $\mathcal{B}$ **do** :
$z_A := z_A - \varepsilon$ et $\zeta_{A'} := \zeta_{A'} - \varepsilon$ pour tout $A' \in \mathcal{B}$ avec $A' \subseteq A$.
$\Delta := \Delta + \varepsilon$.

⑨ **While** il existe $A \in \mathcal{B}$ interne maximal avec $z_A = 0$ et $|A| > 1$ **do** :
(«défaire»)
$\mathcal{B} := \mathcal{B} \setminus \{A\}$.
Soit $y := \sigma(\rho^{k_v}(v))$ pour un sommet $v \in A$.
Soit $Q(y) :=$ BLOSSOMPATH(y) donné par
$(y = r_0, r_1, r_2, \ldots, r_{2l-1}, r_{2l} = \rho^{k_y}(y))$.
Marquer tout $v \in A$ avec $\rho^{k_v-1}(v) \notin V(Q(y))$ comme hors de la forêt.
Marquer tout $v \in A$ avec $\rho^{k_v-1}(v) = r_{2i-1}$ pour un i comme externe.
For tout $v \in A$ avec $\rho^{k_v-1}(v) = r_{2i}$ pour un i (v reste interne) **do** :
$\sigma(\rho^{k_v}(v)) := r_j$ et $\chi(r_j) := r_{j-1}$, où
$j := \min\{j' \in \{0, \ldots, 2l\} : \rho^{k_{r_{j'}}-1}(r_{j'}) = \rho^{k_v-1}(v)\}$.
For tout $v \in A$ **do** : $k_v := k_v - 1$.
For chaque nouveau sommet hors de la forêt v **do** : Soit τ_v un sommet externe w tel que $c((v,w)) - \zeta_{\{v\}} - \zeta_{\{w\}}$ soit minimum.
For chaque nouveau sommet externe v **do** : UPDATE(v).
Go to ③.

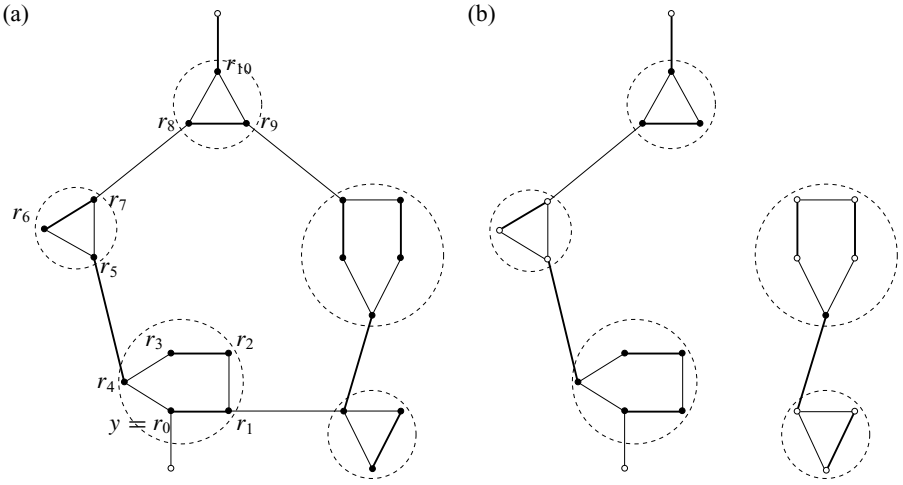


Figure 11.3.

Notons qu'ici il est possible que $\varepsilon = 0$. D'autre part nous n'avons pas besoin de connaître les variables τ_v^A explicitement. L'étape «défaire» ⑨ est illustrée par la figure 11.3, qui représente un blossom avec dix-neuf sommets qui est défait. Deux des cinq sous-blossoms deviennent hors de la forêt, deux deviennent internes et un devient externe.

Dans ⑥, on doit trouver les composantes connexes de la forêt de blossoms F. Cela se fait en temps linéaire par la proposition 2.17.

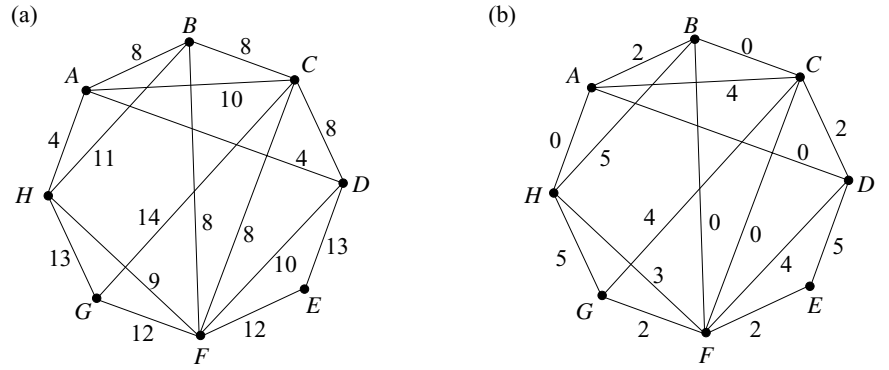


Figure 11.4.

Avant d'étudier la complexité de cet algorithme, illustrons ses phases principales à partir d'un exemple. Étudions le graphe de la figure 11.4(a). La solution initiale est : $z_{\{a\}} = z_{\{d\}} = z_{\{h\}} = 2$, $z_{\{b\}} = z_{\{c\}} = z_{\{f\}} = 4$ et $z_{\{e\}} = z_{\{g\}} = 6$. Les écarts sont montrés sur la figure 11.4(b). Les arêtes serrées sont donc initialement $(a, d), (a, h), (b, c), (b, f), (c, f)$.

Nous supposerons que l'algorithme balaye l'ensemble des sommets dans l'ordre alphabétique. Les premières étapes sont alors

$$\text{augmenter}(a, d), \qquad \text{augmenter}(b, c), \qquad \text{grossir}(f, b).$$

La figure 11.5(a) décrit la forêt de blossoms à l'itération courante.

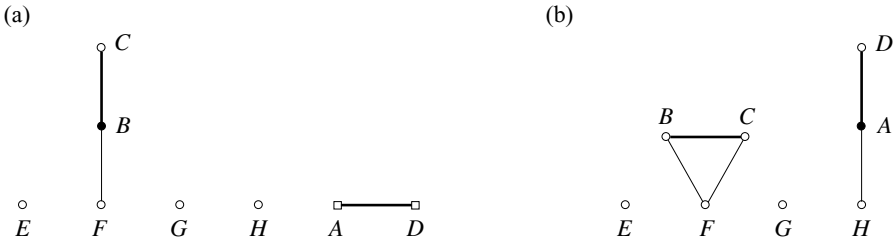


Figure 11.5.

Les étapes suivantes sont :

$$\text{contracter}(f, c), \qquad \text{grossir}(h, a),$$

et produisent la forêt de blossoms représentée sur la figure 11.5(b). Toutes les arêtes serrées ont été examinées, et il nous faut changer la solution duale. Nous exécutons ⑧ et calculons $\varepsilon = \varepsilon_3 = 1$ en posant $A = \{b, c, f\}$ et $\tau_d^A = d$. Les nouvelles variables duales sont $z_{\{b,c,f\}} = 1$, $z_{\{a\}} = 1$, $z_{\{d\}} = z_{\{h\}} = 3$, $z_{\{b\}} = z_{\{c\}} =$

$z_{\{f\}} = 4$, $z_{\{e\}} = z_{\{g\}} = 7$. Les nouveaux écarts sont montrés sur la figure 11.6(a). L'étape suivante est

$$\text{augmenter}(d, c).$$

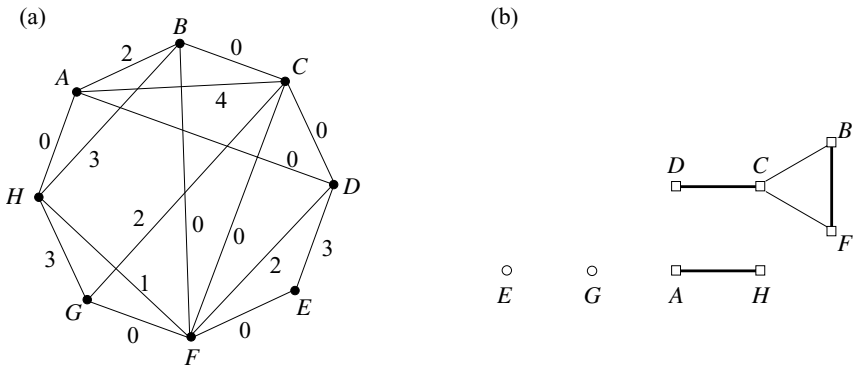


Figure 11.6.

Le blossom $\{b, c, f\}$ devient hors de la forêt (figure 11.6(b)). L'arête (e, f) est maintenant serrée, mais dans le changement dual précédent nous savons que $balayage(d) :=$ *faux*. Nous devons donc exécuter deux fois ⑧ avec $\varepsilon = \varepsilon_3 = 0$ avant de démarrer les étapes suivantes :

$$\text{grossir}(e, f), \qquad \text{grossir}(d, a).$$

Nous en arrivons à la figure 11.7(a).

Il n'existe plus d'arête serrée incidente à un sommet externe, donc nous exécutons une fois de plus ⑧. Nous avons $\varepsilon = \varepsilon_1 = 1$ et nous obtenons la nouvelle solution duale $z_{\{b,c,f\}} = 0$, $z_{\{a\}} = 0$, $z_{\{d\}} = z_{\{h\}} = z_{\{b\}} = z_{\{c\}} = z_{\{f\}} = 4$, $z_{\{e\}} = z_{\{g\}} = 8$. Les nouveaux écarts sont indiqués sur la figure 11.7(b). Puisque la variable duale associée au blossom interne $\{B, C, F\}$ devient nulle, nous devons

$$\text{défaire}(\{b, c, f\}).$$

Nous obtenons la forêt générale de blossoms de la figure 11.8(a). Après une nouvelle modification du dual avec $\varepsilon = \varepsilon_3 = \frac{1}{2}$ nous obtenons $z_{\{a\}} = -0,5$, $z_{\{c\}} = z_{\{f\}} = 3,5$, $z_{\{b\}} = z_{\{d\}} = z_{\{h\}} = 4,5$, $z_{\{e\}} = z_{\{g\}} = 8,5$ (les écarts sont indiqués sur la figure 11.8(b)). Les étapes finales sont :

$$\text{contacter}(d, e), \qquad \text{augmenter}(g, h),$$

et l'algorithme se termine. Le couplage obtenu à la fin est $M = \{(e, f), (b, c), (a, d), (g, h)\}$. Nous vérifions que son poids, 37, est égal à la somme des variables duales.

Vérifions que l'algorithme s'exécute correctement.

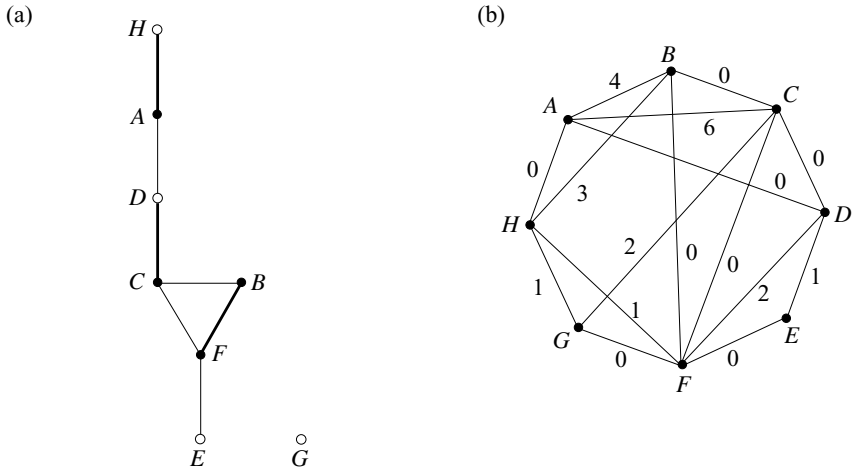


Figure 11.7.

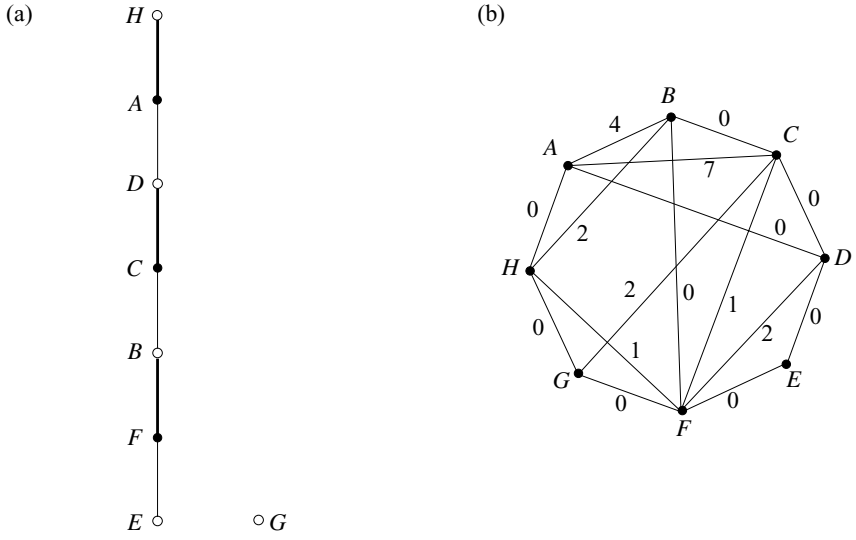


Figure 11.8.

Proposition 11.6. *Les conditions suivantes sont vérifiées à chaque étape de l'*ALGORITHME DU COUPLAGE AVEC POIDS *:*

(a) *$\mathcal{B}$ est une famille laminaire. $\mathcal{B} = \big\{\{v \in V(G) : b^i(v) = j$ pour un indice $i\} : j = 1, \ldots, B\big\}$. Les ensembles $V_{\rho^{k_r}(r)} := \{v : \rho^{k_v}(v) = \rho^{k_r}(r)\}$ sont exactement les éléments maximaux de $\mathcal{B}$. Les sommets dans chaque V_r sont marqués comme étant externes, internes ou hors de la forêt. Chaque $(V_r, \{(v, \varphi^{k_v}(v)) : v \in V_r \setminus \{r\}\} \cup \{(v, \mu(v)) : v \in V_r \setminus \{r\}\})$ est un blossom de base r.*

(b) *L'ensemble* $(x, \mu(x))$ *est un couplage* M. M *contient un couplage presque parfait à l'intérieur de chaque élément de* $\mathcal{B}$.

(c) *Si* $b \in \{1, \ldots, K\}$ *soit* $X(b) := \{v \in V(G) : b^i(v) = b \text{ pour un indice } i\}$. *Alors les variables* $\mu(v)$ *et* $\varphi^i(v)$, *pour les* v *et les* i *tels que* $b^i(v) = b$, *sont associées à une décomposition en oreilles* M*-alternées dans* $G[X(b)]$.

(d) *Les arêtes* $(x, \mu(x))$ *et* $(x, \varphi^i(x))$ *pour tout* x *et tout* i, *et les arêtes* $(\sigma(x), \chi(\sigma(x)))$ *pour toutes les bases* x *des blossoms internes maximaux, sont serrées.*

(e) *Les arêtes* $(x, \mu(x))$, $(x, \varphi^{k_x}(x))$ *pour les sommets internes ou externes* x, *ainsi que les arêtes* $(\sigma(x), \chi(\sigma(x)))$ *pour les bases* x *des blossoms internes maximaux, induisent une forêt générale de blossoms* F *par rapport à* M. *Les marquages (interne, externe, hors de la forêt) sont consistants avec* F.

(f) *La contraction des sous-blossoms maximaux d'un blossom* $\mathcal{B}$ *est un cycle.*

(g) *Pour chaque sommet externe* v, *la procédure* TREEPATH *fournit une chaîne* M*-alternée de* v *à* r, *où* r *est la racine de l'arbre de* F *contenant* v.

Preuve. Ces propriétés sont clairement vérifiées (après la première exécution de ②). Montrons qu'elles restent vérifiées tout au long de l'algorithme. C'est vrai pour (a) en considérant ⑦ et ⑨. Pour (b), cela se déduit de la proposition 11.4 et de l'hypothèse que (f) et (g) sont vraies avant l'augmentation.

La preuve que (c) continue à être vraie après la contraction est la même que dans le cas non pondéré (voir lemme 10.30 (c)). Les valeurs φ sont recalculées après augmentation et non modifiées autrement. La condition (d) est vraie par ④.

Il est facile de voir que (e) reste vraie après ⑤ : le blossom contenant y était hors de la forêt et en posant $\chi(y) := x$ et $\sigma(v) := y$ pour sa base v on le rend interne. Le blossom contenant $\mu(\rho^{k_y}(y))$ était hors de la forêt et devient externe.

En ⑥, deux composantes connexes de la forêt générale de blossoms deviennent hors de la forêt et donc (e) reste vraie. En ⑦, les sommets du nouveau blossom deviennent externes puisque r était externe avant. En ⑨, pour les sommets $v \in A$ tels que $\rho^{k_v-1}(v) \notin V(Q(y))$, nous avons également $\mu(\rho^{k_v}(v)) \notin V(Q(y))$; ces sommets deviennent donc hors de la forêt. Pour tout autre $v \in A$, $\rho^{k_v-1}(v) = r_k$ pour un certain k. Puisque $(r_i, r_{i+1}) \in M$ si et seulement si i est pair, v devient externe si et seulement si k est impair.

(f) est vrai pour tout blossom, puisque chaque nouveau blossom est issu d'un cycle impair dans ⑦. Pour voir que (g) reste vraie, il suffit d'observer que $\sigma(x)$ et $\chi(\sigma(x))$ sont correctement définis pour toutes les bases x des blossoms internes maximaux. Cela se vérifie facilement pour ⑤ et pour ⑨. □

La proposition 11.6(a) justifie l'appellation interne, externe, hors de la forêt pour les éléments maximaux de $\mathcal{B}$ dans les procédures ⑧ et ⑨ de l'algorithme.

Montrons maintenant qu'au cours de l'algorithme la solution du dual est toujours réalisable.

Lemme 11.7. *À toute étape de l'algorithme,* z *est une solution duale réalisable. Si* $\varepsilon = \infty$, *alors* G *n'a pas de couplage parfait.*

Preuve. Nous avons toujours $z_A = 0$ pour tout $A \in \mathcal{A} \setminus \mathcal{B}$. z_A décroît seulement si $A \in \mathcal{B}$ est maximal dans $\mathcal{B}$ et est interne. Donc le choix de ε_1 nous assure que z_A continuera à être non négatif pour tout A avec $|A| > 1$.

Comment les contraintes $\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A \leq c(e)$ peuvent-elles être violées ? Si $\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A$ augmente dans ⑧, e connecte un sommet externe à un sommet hors de la forêt ou connecte deux blossoms distincts. Donc la valeur maximum de ε qui assure que la nouvelle solution z satisfait $\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A \leq c(e)$ est $écart(e)$ dans le premier cas et $\frac{1}{2} écart(e)$ dans le second.

Nous devons donc montrer que ε_2 et ε_3 sont correctement calculés :

$$\varepsilon_2 \;=\; \min\{écart(v,w) : v \text{ externe},\ w \text{ hors de la forêt}\}$$

et

$$\varepsilon_3 \;=\; \frac{1}{2}\min\{écart(v,w) : v, w \text{ externe},\ \rho^{k_v}(v) \neq \rho^{k_w}(w)\}.$$

Pour ε_2 cela est facile à voir, puisque pour tout sommet hors de la forêt v τ_v est toujours le sommet externe w qui minimise $écart(v,w) = c((v,w)) - \zeta_{\{v\}} - \zeta_{\{w\}}$.

Pour ε_3, montrons qu'à chaque étape de l'algorithme les conditions suivantes sont vérifiées pour tout v et tout $A \in \mathcal{B}$ tel qu'il n'existe aucun $\bar{A} \in \mathcal{B}$ tel que $A \cup \{v\} \subseteq \bar{A}$:

(a) $\tau_v^A \in A$.

(b) $écart(v, \tau_v^A) = \min\{écart(v,u) : u \in A\}$.

(c) $\zeta_A = \sum_{B \in \mathcal{B}: A \subseteq B} z_B$. Δ est la somme des valeurs ε dans tous les changements précédents du dual.

(d) $écart(v, \tau_v^A) = t_v^A - \Delta - \zeta_A$.

(e) $t^A = \min\{t_v^A : v \text{ externe et il n'existe aucun } \bar{A} \in \mathcal{B} \text{ avec } A \cup \{v\} \subseteq \bar{A}\}$.

On voit facilement que (a), (c) (e) sont vraies. (b) et (d) sont vraies quand τ_v^A est défini (en ⑦ ou dans UPDATE(v)), et ensuite $écart(v,u)$ décroît de la même quantité que celle de l'accroissement de $\Delta + \zeta_A$ (par (c)). (a), (b), (d), et (e) impliquent que ε_3 est correctement calculé.

Supposons que $\varepsilon = \infty$, c.-à-d. que ε puisse être choisi arbitrairement grand sans détruire la réalisabilité du dual. Puisque la fonction objectif $\mathbb{1}z$ augmente d'au moins ε dans ⑧, le dual est non borné (11.2), ce qui implique que le problème primal (11.1) n'est pas réalisable par le théorème 3.27. □

Prouvons alors que l'algorithme s'exécute correctement :

Théorème 11.8. *Si l'algorithme termine en ⑥, l'ensemble des arêtes $(x, \mu(x))$ et un couplage parfait de poids minimum dans G.*

Preuve. Soit x le vecteur d'incidence de M (le couplage constitué des arêtes $(x, \mu(x))$). Les conditions des écarts complémentaires

$$x_e > 0 \quad \Rightarrow \quad \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e)$$
$$z_A > 0 \quad \Rightarrow \quad \sum_{e \in \delta(A)} x_e = 1$$

sont vérifiées : les premières parce que toute arête du couplage est serrée (proposition 11.6(d)), et les secondes par la proposition 11.6(b).

Puisque les solutions du primal et du dual sont réalisables (lemme 11.7), elles sont toutes deux optimales (corollaire 3.23). Donc x est optimal pour le PL (11.1) et entier, ce qui prouve que M est un couplage parfait de poids minimum. □

Nous n'avons pas encore démontré que l'algorithme se termine.

Théorème 11.9. *Le temps de calcul de l'*ALGORITHME DU COUPLAGE AVEC POIDS *entre deux augmentations est $O(n^2)$. Le temps total de calcul est $O(n^3)$.*

Preuve. Par le lemme 11.5 et la proposition 11.6(a), la procédure UPDATE s'exécute en temps linéaire.

Le temps de calcul de ② et ⑥ (qui s'exécutent une seule fois à chaque augmentation) est $O(n^2)$.

Chaque étape ⑤, ⑦, et ⑨ se fait en un temps $O(nk)$ où k est le nombre de nouveaux sommets externes. (En ⑦, le nombre de sous-ensembles propres maximaux A' de A à considérer est au plus $2k+1$: chaque second sous-blossom d'un nouveau blossom était nécessairement interne.) Puisque tout nouveau sommet externe continuera à être externe jusqu'à l'augmentation suivante, le temps total d'exécution de ⑤, ⑦ et ⑨ entre deux augmentations est $O(n^2)$.

Il reste à estimer le temps de calcul de ⑧, ③ et ④. Supposons que $\varepsilon \neq \varepsilon_1$ en ⑧. Une nouvelle arête serrée est créée en ⑧ à cause des variables t_v et t_v^A. Nous poursuivons en ③ et ④ où, après un temps au plus $O(n)$, cette arête est vérifiée. Puisqu'elle connecte un sommet externe à un sommet hors de la forêt ou deux composantes connexes externes, nous pouvons appliquer une des trois procédures ⑤, ⑥, ⑦. Si $\varepsilon = \varepsilon_1$, nous devons appliquer ⑨.

Cela montre que le nombre de fois où ⑧ est exécuté est inférieur ou égal au nombre de fois où une des procédures ⑤, ⑥, ⑦, ⑨ est exécutée. Puisque le temps de calcul de ⑧ est $O(n)$, la borne $O(n^2)$ entre deux augmentations est démontrée. Notons que nous n'excluons pas le cas $\varepsilon = 0$.

Puisque le nombre d'augmentations est $\frac{n}{2}$, le temps de calcul est $O(n^3)$. □

Corollaire 11.10. *Le* PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM *peut se résoudre en un temps $O(n^3)$.*

Preuve. Cela est une conséquence des théorèmes 11.8 et 11.9. □

La première implémentation en $O(n^3)$ de l'algorithme d'Edmonds pour le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM a été réalisée par Gabow [1973] (voir aussi Gabow [1976] et Lawler [1976]). Le meilleur temps de calcul théorique, qui est $O(mn + n^2 \log n)$, a également été obtenu par Gabow [1990].

Pour les graphes planaires, un couplage parfait de poids minimum peut se trouver en $O\left(n^{\frac{3}{2}} \log n\right)$, comme l'ont montré Lipton et Tarjan [1979,1980] par une approche diviser pour régner, utilisant le fait qu'un graphe planaire a de petits «séparateurs». Pour les instances euclidiennes (graphe complet dont les sommets sont des points dans le plan et dont les coûts sont les distances euclidiennes entre les paires de sommets), Varadarajan [1998] a trouvé un algorithme en $O\left(n^{\frac{3}{2}} \log^5 n\right)$.

Les implémentations actuelles probablement les plus efficaces ont été proposées par Mehlhorn et Schäfer [2000] et Cook et Rohe [1999]. Ils résolvent des problèmes de couplage avec plusieurs millions de sommets de manière optimale. Un algorithme «primal» de couplage avec poids qui maintient toujours un couplage parfait et qui obtient une solution duale réalisable à la fin a été décrit par Cunningham et Marsh [1978].

11.4 Postoptimalité

Nous prouverons dans ce paragraphe deux résultats de postoptimalité que nous utiliserons au paragraphe 12.2.

Lemme 11.11. (Weber [1981], Ball et Derigs [1983]) *Supposons que nous ayons exécuté l'*ALGORITHME DU COUPLAGE AVEC POIDS *sur une instance* (G, c). *Soit* $s \in V(G)$, *et soit* $c' : E(G) \to \mathbb{R}$ *tel que* $c'(e) = c(e)$ *pour tout* $e \notin \delta(s)$. *On peut trouver un couplage parfait de poids minimum par rapport à* (G, c') *en un temps* $O(n^2)$.

Preuve. Soit $t := \mu(s)$. Si s n'est contenu dans aucun blossom non trivial, c.-à-d. $k_s = 0$, alors la première étape consiste simplement à poser $\mu(s) := s$ et $\mu(t) := t$. Sinon, il faut défaire tous les blossoms contenant s. Pour cela , nous effectuons des changements du dual d'une valeur totale $\sum_{A:\, s\in A,\, |A|>1} z_A$ et pendant toutes ces modifications s restera interne. Considérons la construction suivante :

Soit $V(G) := V(G) \,\dot\cup\, \{a, b\}$ et $E(G) := E(G) \cup \{(a, s), (b, t)\}$.
Soit $c((a, s)) := \zeta_{\{s\}}$ et $c((b, t)) := 2 \sum_{A:\, s\in A,\, |A|>1} z_A + \zeta_{\{t\}}$.
Soit $\mu(a) := a$ et $\mu(b) := b$. Marquer a et b comme sommets externes.
Soit $\mathcal{B} := \mathcal{B} \cup \{\{a\}, \{b\}\}$, $z_{\{a\}} := 0$, $z_{\{b\}} := 0$, $\zeta_{\{a\}} := 0$, $\zeta_{\{b\}} := 0$.
Soit $k_a := 0$, $k_b := 0$, $\rho^0(a) := a$, $\rho^0(b) := b$, $\varphi^0(a) := a$, $\varphi^0(b) := b$.
UPDATE(a). UPDATE(b).

Nous pouvons poursuivre l'exécution de l'algorithme avec l'instance (G, c) en partant de son output, pour traiter l'exemple modifié (le graphe augmenté de deux sommets et de deux arêtes). En particulier, la solution duale z optimale est réalisable pour l'exemple modifié et l'arête (a, s) est serrée. Posons $balayage(a) :=$ *faux* et poursuivons l'algorithme en démarrant en ③. L'algorithme grossira la forêt à partir de (a, s), et s deviendra interne.

Par le théorème 11.9 l'algorithme se termine par une augmentation après $O(n^2)$ étapes. Or a, s, t, b est la seule chaîne augmentante. Donc l'arête (b,t) doit devenir serrée. Au début, $\acute{e}cart(b,t) = 2\sum_{A\in\mathcal{A},\, s\in A,\, |A|>1} z_A$.

Le sommet s restera tout le temps interne. Donc $\zeta_{\{s\}}$ diminuera à chaque modification du dual. À la fin, tous les blossoms A contenant s sont défaits. Il nous suffit alors de supprimer les sommets a, b et les arêtes (a,s) et (b,t), et poser $\mathcal{B} := \mathcal{B} \setminus \{\{a\},\{b\}\}$ et $\mu(s) := s$, $\mu(t) := t$.

s et t sont maintenant externes et il n'y a pas de sommets internes ; de plus, aucune arête incidente à s n'appartient à la forêt générale de blossoms. Nous pouvons aisément changer les poids des arêtes incidentes à s aussi bien que $z_{\{s\}}$, pourvu que nous maintenions la réalisabilité du dual. Cela se fait facilement : nous calculons d'abord les écarts relatifs aux nouveaux poids et en ajoutant à $z_{\{s\}}$ la quantité $\min_{e\in\delta(s)} \acute{e}cart(e)$. Nous posons $balayage(s) :=$ *faux* et nous poursuivons l'algorithme à partir de ③. Par le théorème 11.9, l'algorithme se termine après $O(n^2)$ étapes avec un couplage parfait de poids minimum relatif aux nouveaux poids. □

Un résultat analogue avec une approche «primale» a été trouvé par Cunningham et Marsh [1978]. Le lemme suivant traite de l'addition de deux sommets à une instance qui a déjà été résolue.

Lemme 11.12. *Soit (G,c) une instance du* PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM*, et soient $s,t \in V(G)$. Supposons que l'*ALGORITHME DU COUPLAGE AVEC POIDS *ait résolu l'instance $(G - \{s,t\}, c)$. On peut trouver un couplage parfait de poids minimum sur l'instance (G,c) en un temps $O(n^2)$.*

Preuve. L'addition de deux sommets requiert l'initialisation de structures de données comme dans le cas précédent. La variable duale z_v est choisie de telle sorte que $\min_{e\in\delta(v)} \acute{e}cart(e) = 0$ (pour $v \in \{s,t\}$). Nous posons ensuite $balayage(s) := balayage(t) :=$ *faux* et nous redémarrons l'ALGORITHME DE COUPLAGE AVEC POIDS en ③. □

11.5 Polytope du couplage

Un résultat dérivé de l'ALGORITHME DE COUPLAGE AVEC POIDS est la caractérisation d'Edmonds du polytope des couplages parfaits. Nous utiliserons de nouveau la notation $\mathcal{A} := \{A \subseteq V(G) : |A| \text{ impair}\}$.

Théorème 11.13. (Edmonds [1965]) *Soit G un graphe non orienté. Le* **polytope des couplages parfaits** *de G, c.-à-d. l'enveloppe convexe des vecteurs d'incidence de tous les couplages parfaits de G, est l'ensemble des vecteurs x vérifiant*

$$\begin{aligned} x_e &\geq 0 && (e \in E(G)) \\ \sum_{e\in\delta(v)} x_e &= 1 && (v \in V(G)) \\ \sum_{e\in\delta(A)} x_e &\geq 1 && (A \in \mathcal{A}). \end{aligned}$$

Preuve. Par le corollaire 3.32, il suffit de montrer que tous les sommets du polytope de l'énoncé sont entiers. Par le théorème 5.13 cela est vrai si le problème de minimisation a une solution entière quand les poids sont entiers. Mais c'est précisément ce que fait l'ALGORITHME DU COUPLAGE AVEC POIDS (voir démonstration du theorème 11.8). □

Une autre preuve sera donnée au paragraphe 12.3 (voir remarque après le théorème 12.16).

Nous pouvons également caractériser le **polytope du couplage**, c.-à-d. l'enveloppe convexe des vecteurs d'incidence de tous les couplages d'un graphe non orienté G :

Théorème 11.14. (Edmonds [1965]) *Soit G un graphe. Le polytope du couplage de G est l'ensemble des vecteurs $x \in \mathbb{R}_+^{E(G)}$ qui vérifient*

$$\sum_{e\in\delta(v)} x_e \leq 1 \text{ pour } v \in V(G) \quad \text{et} \quad \sum_{e\in E(G[A])} x_e \leq \frac{|A|-1}{2} \text{ pour } A \in \mathcal{A}.$$

Preuve. Puisque le vecteur d'incidence de tout couplage satisfait à l'évidence ces inégalités, nous devons seulement montrer que tout vecteur $x \in \mathbb{R}_+^{E(G)}$ qui vérifie $\sum_{e\in\delta(v)} x_e \leq 1$ pour $v \in V(G)$ et $\sum_{e\in E(G[A])} x_e \leq \frac{|A|-1}{2}$ pour $A \in \mathcal{A}$ est combinaison convexe des vecteurs d'incidence des couplages de G.

Soit H le graphe tel que $V(H) := \{(v,i) : v \in V(G),\ i \in \{1,2\}\}$, et $E(H) := \{((v,i),(w,i)) : (v,w) \in E(G),\ i \in \{1,2\}\} \cup \{((v,1),(v,2)) : v \in V(G)\}$. H est constitué de deux copies de G, et les deux copies de chaque sommet sont jointes par une arête. Posons $y_{((v,i),(w,i))} := x_e$ pour tout $e = (v,w) \in E(G)$ et $i \in \{1,2\}$, et $y_{((v,1),(v,2))} := 1 - \sum_{e\in\delta_G(v)} x_e$ pour tout $v \in V(G)$. Montrons que y appartient au polytope des couplages parfaits de H. En considérant alors le sous-graphe induit par $\{(v,1) : v \in V(G)\}$, qui est isomorphe à G, x sera bien combinaison convexe des vecteurs d'incidence des couplages de G.

Il est clair que $y \in \mathbb{R}_+^{E(H)}$ et que $\sum_{e\in\delta_H(v)} y_e = 1$ pour tout $v \in V(H)$. Pour montrer que y appartient au polytope des couplages parfaits de H, nous nous servirons du théorème 11.13. Soit donc $X \subseteq V(H)$ avec $|X|$ impair. Montrons que $\sum_{e\in\delta_H(X)} y_e \geq 1$. Soit $A := \{v \in V(G) : (v,1) \in X,\ (v,2) \notin X\}$, $B := \{v \in V(G) : (v,1) \in X,\ (v,2) \in X\}$ et $C := \{v \in V(G) : (v,1) \notin X,\ (v,2) \in X\}$. Puisque $|X|$ est impair, A ou C est de cardinalité impaire ; on peut toujours supposer $|A|$ impair. Posons $A_i := \{(a,i) : a \in A\}$ et $B_i := \{(b,i) : b \in B\}$ pour $i = 1,2$ (voir figure 11.9). Alors

$$\begin{aligned}
\sum_{e\in\delta_H(X)} y_e &\geq \sum_{v\in A_1} \sum_{e\in\delta_H(v)} y_e - 2 \sum_{e\in E(H[A_1])} y_e - \sum_{e\in E_H(A_1,B_1)} y_e + \sum_{e\in E_H(B_2,A_2)} y_e \\
&= \sum_{v\in A_1} \sum_{e\in\delta_H(v)} y_e - 2 \sum_{e\in E(G[A])} x_e \\
&\geq |A_1| - (|A|-1) = 1.
\end{aligned}$$

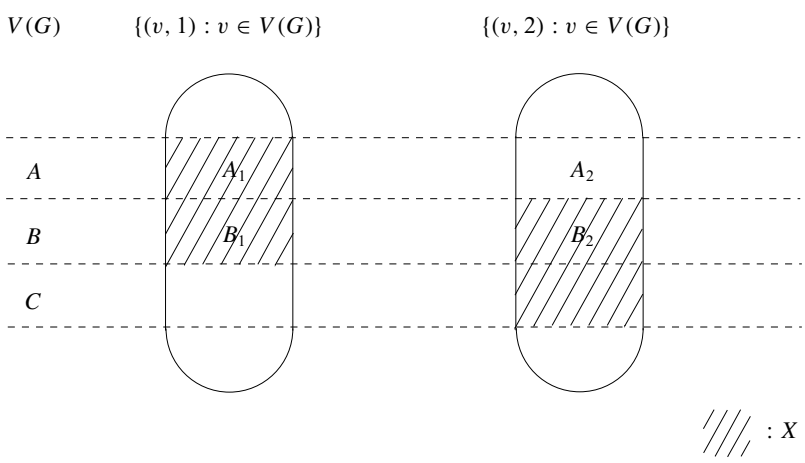


Figure 11.9.

□

Nous pouvons également prouver un résultat plus général :

Théorème 11.15. (Cunningham et Marsh [1978]) *Si G est un graphe non orienté, le système d'inégalités*

$$\begin{array}{rcll} x_e & \geq & 0 & (e \in E(G)) \\ \sum_{e \in \delta(v)} x_e & \leq & 1 & (v \in V(G)) \\ \sum_{e \subseteq A} x_e & \leq & \frac{|A|-1}{2} & (A \in \mathcal{A},\ |A| > 1) \end{array}$$

est TDI.

Preuve. Considérons le PL : $\max \sum_{e \in E(G)} c(e) x_e$ avec les contraintes précédentes. Le dual est :

$$\begin{array}{lrcll} \min & \sum_{v \in V(G)} y_v + \sum_{A \in \mathcal{A},\, |A|>1} \frac{|A|-1}{2} z_A & & & \\ \text{s.c.} & \sum_{v \in e} y_v + \sum_{A \in \mathcal{A},\, e \subseteq A} z_A & \geq & c(e) & (e \in E(G)) \\ & y_v & \geq & 0 & (v \in V(G)) \\ & z_A & \geq & 0 & (A \in \mathcal{A},\ |A| > 1) \end{array}$$

Soit (G, c) le plus petit contre-exemple, c.-à-d. (G, c) n'a pas de solution duale optimale entière et $|V(G)|+|E(G)|+\sum_{e \in E(G)} |c(e)|$ est minimum. Alors $c(e) \geq 1$

pour tout e (sinon nous pourrions éliminer une arête de poids négatif), et G n'a pas de sommet isolé (sinon nous pourrions l'enlever).

De plus, pour toute solution optimale y, z, nous affirmons que $y = 0$. Supposons en effet qu'il existe $v \in V(G)$ avec $y_v > 0$. Les conditions des écarts complémentaires (corollaire 3.23) impliquent que $\sum_{e \in \delta(v)} x_e = 1$ pour toute solution primale optimale x. Mais si on soustrait 1 à $c(e)$ pour tout $e \in \delta(v)$, on obtient une plus petite instance (G, c'), dont la valeur est diminuée de 1 (nous utilisons ici la propriété d'intégralité du primal, c.-à-d. le théorème 11.14). Puisque (G, c) est le plus petit contre-exemple, il existe une solution duale optimale entière y', z' pour (G, c'). En ajoutant 1 à y'_v, on obtient une solution optimale du dual pour (G, c), ce qui est une contradiction.

Soit alors $y = 0$, z une solution optimale du dual telle que

$$\sum_{A \in \mathcal{A},\, |A|>1} |A|^2 z_A \tag{11.4}$$

soit le plus grand possible. Nous affirmons que $\mathcal{F} := \{A : z_A > 0\}$ est laminaire. Supposons en effet qu'il existe $X, Y \in \mathcal{F}$ avec $X \setminus Y \neq \emptyset$, $Y \setminus X \neq \emptyset$ et $X \cap Y \neq \emptyset$. Soit $\epsilon := \min\{z_X, z_Y\} > 0$.

Si $|X \cap Y|$ est impair, $|X \cup Y|$ est aussi impair. Posons $z'_X := z_X - \epsilon$, $z'_Y := z_Y - \epsilon$, $z'_{X\cap Y} := z_{X\cap Y} + \epsilon$ (sauf si $|X \cap Y| = 1$), $z'_{X\cup Y} := z_{X\cup Y} + \epsilon$ et $z'_A := z_A$ pour tous les autres ensembles A. y, z' est une solution duale réalisable qui de plus est aussi optimale ; c'est une contradiction puisque (11.4) est plus grand.

Si $|X \cap Y|$ est pair, alors $|X \setminus Y|$ et $|Y \setminus X|$ sont impairs. Soit $z'_X := z_X - \epsilon$, $z'_Y := z_Y - \epsilon$, $z'_{X\setminus Y} := z_{X\setminus Y} + \epsilon$ (sauf si $|X \setminus Y| = 1$), $z'_{Y\setminus X} := z_{Y\setminus X} + \epsilon$ (sauf si $|Y \setminus X| = 1$) et $z'_A := z_A$ pour tous les autres ensembles A. Posons $y'_v := y_v + \epsilon$ pour $v \in X \cap Y$ et $y'_v := y_v$ pour $v \notin X \cap Y$. Alors y', z' est une solution duale réalisable qui est aussi optimale. Cela contredit le fait que pour toute solution optimale du dual nous avons $y = 0$.

Soit alors $A \in \mathcal{F}$ avec $z_A \notin \mathbb{Z}$ et A maximal. Posons $\epsilon := z_A - \lfloor z_A \rfloor > 0$. Soient $A_1, \ldots, A_k$ les sous-ensembles propres maximaux de A dans $\mathcal{F}$; ils sont disjoints puisque $\mathcal{F}$ est laminaire. En posant $z'_A := z_A - \epsilon$ et $z'_{A_i} := z_{A_i} + \epsilon$ pour $i = 1, \ldots, k$ (et $z'_D := z_D$ pour tous les autres ensembles $D \in \mathcal{A}$), nous obtenons une autre solution duale réalisable $y = 0, z'$ (puisque c est entier). Il vient

$$\sum_{B \in \mathcal{A},\, |B|>1} \frac{|B|-1}{2} z'_B < \sum_{B \in \mathcal{A},\, |B|>1} \frac{|B|-1}{2} z_B,$$

contredisant l'optimalité de la solution du dual originale $y = 0, z$. □

Cette preuve est due à Schrijver [1983a]. Pour d'autres preuves, voir Lovász [1979] et Schrijver [1983b]. Ces dernières n'utilisent pas le théorème 11.14. De plus, en remplaçant $\sum_{e \in \delta(v)} x_e \leq 1$ par $\sum_{e \in \delta(v)} x_e = 1$ pour $v \in V(G)$ dans le théorème 11.15 on obtient une autre description du polytope des couplages parfaits, qui est également TDI (par le théorème 5.18). Le théorème 11.13 peut facilement se déduire de cela ; cependant, le système d'inégalités linéaires du théorème

11.13 n'est pas TDI en général (K_4 est un contre-exemple). Le théorème 11.15 implique également la formule de Berge-Tutte (théorème 10.14 ; voir exercice 14). Des généralisations seront présentées dans le paragraphe 12.1.

Exercices

1. Utiliser le théorème 11.2 pour démontrer une version pondérée du théorème de König 10.2.
   (Egerváry [1931])
2. Décrire l'enveloppe convexe des vecteurs d'incidence des
   (a) couvertures par les sommets,
   (b) ensembles stables,
   (c) couvertures par des arêtes,
   d'un graphe biparti G. Montrer comment en déduire le théorème 10.2 et la condition de l'exercice 2(c) du chapitre 10.
   Indication : utiliser le théorème 5.25 et le corollaire 5.21.
3. Prouver le théorème de Birkhoff-von-Neumann 11.3 directement.
4. Soit G un graphe et soit P le polytope fractionnaire des couplages parfaits de G. Montrer que les sommets de P sont précisément les vecteurs x tels que
$$x_e = \begin{cases} \frac{1}{2} & \text{si } e \in E(C_1) \cup \cdots \cup E(C_k) \\ 1 & \text{si } e \in M \\ 0 & \text{sinon} \end{cases},$$
   $C_1, \ldots, C_k$ étant des cycles sommet-disjoints impairs et M étant un couplage parfait dans $G - (V(C_1) \cup \cdots \cup V(C_k))$.
   (Balinski [1972] ; voir Lovász [1979])
5. Soit G un graphe biparti ayant une bipartition $V = A \dot\cup B$ avec $A = \{a_1, \ldots, a_p\}$, $B = \{b_1, \ldots, b_q\}$. Soit $c : E(G) \to \mathbb{R}$ un ensemble de poids sur les arêtes. Nous recherchons un couplage M qui préserve l'ordre des poids, ce qui signifie : si $(a_i, b_j), (a_{i'}, b_{j'}) \in M$ avec $i < i'$, alors $j < j'$. Résoudre ce problème par un algorithme en $O(n^3)$.
   Indication : utiliser la programmation dynamique.
6. Montrer que $|\mathcal{B}| \leq \frac{3}{2}n$ à chaque étape de l'ALGORITHME DU COUPLAGE PARFAIT DE COÛT MINIMUM.
7. Soit G un graphe muni de poids non négatifs $c : E(G) \to \mathbb{R}_+$. Soit M le couplage à une étape intermédiaire de l'ALGORITHME DE COUPLAGE PARFAIT DE POIDS MINIMUM. Soit X l'ensemble des sommes couverts par M. Montrer que tout couplage couvrant X a un poids supérieur ou égal à celui de M.
   (Ball et Derigs [1983])

8. On dira qu'un graphe avec des poids entiers sur les arêtes a la propriété du cycle pair si le poids de tout cycle est pair. Montrer que l'ALGORITHME DU COUPLAGE PARFAIT DE POIDS MINIMUM appliqué à un graphe avec la propriété du cycle pair conserve cette propriété (par rapport aux écarts sur les arêtes) et conserve également la propriété que la solution du dual est entière. Conclure que dans le cas général il existe une solution optimale du dual z qui est demi-entière (c.-à-d. $2z$ est entier).

9. Quand l'ALGORITHME DU COUPLAGE PARFAIT DE POIDS MINIMUM s'applique aux graphes bipartis, il devient beaucoup plus simple. Repérer les parties de l'algorithme qu'il faut garder et celles qu'on peut supprimer.
Note : on arrive alors à ce qui est connu sous le nom de méthode hongroise pour le PROBLÈME D'AFFECTATION (Kuhn [1955]). Cet algorithme décrit également d'une autre manière la procédure proposée dans la preuve du théorème 11.1.

10. Comment résoudre le problème du couplage à seuil (trouver un couplage parfait M tel que $\max\{c(e) : e \in M\}$ soit minimum) en un temps $O(n^3)$?

11. Montrer comment résoudre le PROBLÈME DE LA COUVERTURE PAR LES ARÊTES DE POIDS MINIMUM en temps polynomial : étant donné un graphe non orienté G et des poids $c : E(G) \to \mathbb{R}$, trouver une couverture par les arêtes de poids minimum.

12. Soit un graphe non orienté G muni de poids $c : E(G) \to \mathbb{R}_+$ et soient deux sommets s et t ; nous recherchons une plus courte chaîne de s à t ayant un nombre pair (impair) d'arêtes. Réduire ce problème à un PROBLÈME DE COUPLAGE PARFAIT DE POIDS MINIMUM.
Indication : prendre deux copies de G, créer une arête de poids nul entre tout sommet et sa copie et supprimer s et t (ou s et la copie de t).
(Grötschel et Pulleyblank [1981])

13. Soit G un graphe k-régulier et $(k-1)$-arête-connexe ayant un nombre pair de sommets, et soit $c : E(G) \to \mathbb{R}_+$. Montrer qu'il existe un couplage parfait M de poids $c(M) \geq \frac{1}{k} c(E(G))$ dans G.
Indication : montrer que $\frac{1}{k}\mathbb{1}$ appartient au polytope des couplages parfaits.

∗ 14. Montrer que le théorème 11.15 implique :

   (a) La formule de Berge-Tutte (théorème 10.14).

   (b) Le théorème 11.13.

   (c) L'existence d'une solution duale optimale demi-entière pour le PL dual (11.2) (voir exercice 8).

   Indication : utiliser le théorème 5.18.

15. Le polytope fractionnaire des couplages parfaits Q de G est identique au polytope des couplages parfaits quand G est biparti (théorème 11.2). Considérons la première troncature de Gomory-Chvátal Q' de Q (définition 5.29). Montrer que Q' est toujours identique au polytope des couplages parfaits.

Références

Littérature générale :

Gerards, A.M.H. [1995] : Matching. In : Handbooks in Operations Research and Management Science ; Volume 7 : Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995, pp. 135–224

Lawler, E.L. [1976] : Combinatorial Optimization ; Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Chapters 5 and 6

Papadimitriou, C.H., Steiglitz, K. [1982] : Combinatorial Optimization ; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Chapter 11

Pulleyblank, W.R. [1995] : Matchings and extensions. In : Handbook of Combinatorics ; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam 1995

Références citées :

Balinski, M.L. [1972] : Establishing the matching polytope. Journal of Combinatorial Theory 13 (1972), 1–13

Ball, M.O., Derigs, U. [1983] : An analysis of alternative strategies for implementing matching algorithms. Networks 13 (1983), 517–549

Birkhoff, G. [1946] : Tres observaciones sobre el algebra lineal. Revista Universidad Nacional de Tucumán, Series A 5 (1946), 147–151

Cook, W., Rohe, A. [1999] : Computing minimum-weight perfect matchings. INFORMS Journal of Computing 11 (1999), 138–148

Cunningham, W.H., Marsh, A.B. [1978] : A primal algorithm for optimum matching. Mathematical Programming Study 8 (1978), 50–72

Edmonds, J. [1965] : Maximum matching and a polyhedron with (0,1) vertices. Journal of Research of the National Bureau of Standards B 69 (1965), 125–130

Egerváry, E. [1931] : Matrixok kombinatorikus tulajdonságairol. Matematikai és Fizikai Lapok 38 (1931), 16–28 [in Hungarian]

Gabow, H.N. [1973] : Implementation of algorithms for maximum matching on non-bipartite graphs. Ph.D. Thesis, Stanford University, Dept. of Computer Science, 1973

Gabow, H.N. [1976] : An efficient implementation of Edmonds' algorithm for maximum matching on graphs. Journal of the ACM 23 (1976), 221–234

Gabow, H.N. [1990] : Data structures for weighted matching and nearest common ancestors with linking. Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (1990), 434–443

Grötschel, M., Pulleyblank, W.R. [1981] : Weakly bipartite graphs and the max-cut problem. Operations Research Letters 1 (1981), 23–27

Kuhn, H.W. [1955] : The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2 (1955), 83–97

Lipton, R.J., Tarjan, R.E. [1979] : A separator theorem for planar graphs. SIAM Journal on Applied Mathematics 36 (1979), 177–189

Lipton, R.J., Tarjan, R.E. [1980] : Applications of a planar separator theorem. SIAM Journal on Computing 9 (1980), 615–627

Lovász, L. [1979] : Graph theory and integer programming. In : Discrete Optimization I ; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, eds.), North-Holland, Amsterdam 1979, pp. 141–158

Mehlhorn, K., Schäfer, G. [2000] : Implementation of $O(nm \log n)$ weighted matchings in general graphs : the power of data structures. In : Algorithm Engineering ; WAE-2000 ; LNCS 1982 (S. Näher, D. Wagner, eds.), pp. 23–38 ; also electronically in The ACM Journal of Experimental Algorithmics 7 (2002)

Monge, G. [1784] : Mémoire sur la théorie des déblais et des remblais. Histoire de l'Académie Royale des Sciences 2 (1784), 666–704

Munkres, J. [1957] : Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5 (1957), 32–38

von Neumann, J. [1953] : A certain zero-sum two-person game equivalent to the optimal assignment problem. In : Contributions to the Theory of Games II ; Ann. of Math. Stud. 28 (H.W. Kuhn, ed.), Princeton University Press, Princeton 1953, pp. 5–12

Schrijver, A. [1983a] : Short proofs on the matching polyhedron. Journal of Combinatorial Theory B 34 (1983), 104–108

Schrijver, A. [1983b] : Min-max results in combinatorial optimization. In : Mathematical Programming ; The State of the Art – Bonn 1982 (A. Bachem, M. Grötschel, B. Korte, eds.), Springer, Berlin 1983, pp. 439–500

Varadarajan, K.R. [1998] : A divide-and-conquer algorithm for min-cost perfect matching in the plane. Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (1998), 320–329

Weber, G.M. [1981] : Sensitivity analysis of optimal matchings. Networks 11 (1981), 41–56

Chapitre 12

b-couplages et *T*-joints

Nous étudierons dans ce chapitre deux autres problèmes d'optimisation combinatoire : le PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM au paragraphe 12.1 et le PROBLÈME DU T-JOINT DE POIDS MINIMUM au paragraphe 12.2. Ces deux problèmes généralisent le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM et incluent d'autres problèmes importants. Mais ils se réduisent également au PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM. Ils se résolvent donc par des algorithmes polynomiaux et possèdent des descriptions polyédrales. Puisque dans les deux cas le PROBLÈME DE SÉPARATION se résout en temps polynomial, il existe un algorithme polynomial (fondé sur la MÉTHODE DES ELLIPSOÏDES ; voir paragraphe 4.6) pour résoudre ces deux généralisations du problème du couplage. Le PROBLÈME DE SÉPARATION se ramène effectivement à la recherche d'une T-coupe minimum dans les deux cas ; voir les paragraphes 12.3 et 12.4. Ce dernier problème : trouver une coupe de capacité minimum $\delta(X)$ quand $|X \cap T|$ est impair pour T ensemble spécifié de sommets se résout avec des techniques de flots.

12.1 *b*-couplages

Définition 12.1. *Soit G un graphe non orienté muni de capacités $u : E(G) \to \mathbb{N} \cup \{\infty\}$ entières et de nombres $b : V(G) \to \mathbb{N}$. Un* ***b*-couplage** *dans (G, u) est une fonction $f : E(G) \to \mathbb{Z}_+$ avec $f(e) \leq u(e)$ pour tout $e \in E(G)$ et $\sum_{e \in \delta(v)} f(e) \leq b(v)$ pour tout $v \in V(G)$. Si $u \equiv 1$, nous dirons que le b-couplage f est* **simple**. *f sera dit* **parfait** *si $\sum_{e \in \delta(v)} f(e) = b(v)$ pour tout $v \in V(G)$.*

Dans le cas $b \equiv 1$, les capacités sont sans objet et nous sommes ramenés au cas des couplages ordinaires. Un b-couplage simple est aussi appelé un b-facteur. Un b-couplage simple peut être vu comme un sous-ensemble d'arêtes. Nous étudierons les 2-couplages simples parfaits au chapitre 21 : ce sont des sous-ensembles d'arêtes tels que chaque sommet soit incident à exactement deux de ces arêtes.

PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM

Instance Un graphe G, des capacités $u : E(G) \to \mathbb{N} \cup \{\infty\}$, des poids $c : E(G) \to \mathbb{R}$, et des nombres $b : V(G) \to \mathbb{N}$.

Tâche Trouver un b-couplage f dans (G, u) de poids $\sum_{e \in E(G)} c(e)f(e)$ maximum.

On peut étendre l'ALGORITHME DU COUPLAGE AVEC POIDS d'Edmonds pour résoudre ce problème (Marsh [1979]). Nous ne décrirons pas cet algorithme ici, mais nous donnerons une description polyédrale qui permet de résoudre le PROBLÈME DE SÉPARATION en temps polynomial. Cela conduit à un algorithme polynomial par la MÉTHODE DES ELLIPSOÏDES (voir corollaire 3.33).

Le **polytope du *b*-couplage** de (G, u) est l'ensemble convexe des vecteurs d'incidence des b-couplages dans (G, u). Étudions d'abord le cas sans capacités :

Théorème 12.2. (Edmonds [1965]) *Soit G un graphe non orienté et $b : V(G) \to \mathbb{N}$. Le polytope du b-couplage de (G, ∞) est l'ensemble des vecteurs $x \in \mathbb{R}_+^{E(G)}$ vérifiant*

$$\begin{aligned} \sum_{e \in \delta(v)} x_e &\leq b(v) && (v \in V(G)); \\ \sum_{e \in E(G[X])} x_e &\leq \left\lfloor \tfrac{1}{2} \sum_{v \in X} b(v) \right\rfloor && (X \subseteq V(G)). \end{aligned}$$

Preuve. Tout b-couplage vérifie manifestement ces contraintes. Soit $x \in \mathbb{R}_+^{E(G)}$ vérifiant $\sum_{e \in \delta(v)} x_e \leq b(v)$ pour $v \in V(G)$ et $\sum_{e \in E(G[X])} x_e \leq \lfloor \frac{1}{2} \sum_{v \in X} b(v) \rfloor$ pour $X \subseteq V(G)$. Montrons que x est combinaison convexe des vecteurs d'incidence de b-couplages.

Construisons un nouveau graphe H obtenu en remplaçant chaque sommet v par $b(v)$ copies : posons $X_v := \{(v, i) : i \in \{1, \ldots, b(v)\}\}$ pour $v \in V(G)$, $V(H) := \bigcup_{v \in V(G)} X_v$ et $E(H) := \{(v', w') : (v, w) \in E(G),\ v' \in X_v,\ w' \in X_w\}$. Soit $y_e := \frac{1}{b(v)b(w)} x_{(v,w)}$ pour tout $e = (v', w') \in E(H)$, $v' \in X_v, w' \in X_w$. Si y est combinaison convexe des vecteurs d'incidence des couplages de H, on voit, en contractant les ensembles X_v ($v \in V(G)$) dans H et en revenant à G et x, que x est combinaison convexe des vecteurs d'incidence des b-couplages dans G.

Pour prouver que y appartient au polytope du couplage de H, nous utiliserons le théorème 11.14. $\sum_{e \in \delta(v)} y_e \leq 1$ est clairement vérifié pour tout $v \in V(H)$. Soit $C \subseteq V(H)$ avec $|C|$ impair. Montrons que $\sum_{e \in E(H[C])} y_e \leq \frac{1}{2}(|C| - 1)$.

Si $X_v \subseteq C$ ou $X_v \cap C = \emptyset$ pour tout $v \in V(G)$, cela se déduit directement des inégalités vérifiées par x. Sinon soit $a, b \in X_v$, $a \in C$, $b \notin C$. Nous avons

$$\begin{aligned}
2\sum_{e\in E(H[C])} y_e &= \sum_{c\in C\setminus\{a\}}\ \sum_{e\in E(\{c\},C\setminus\{c\})} y_e + \sum_{e\in E(\{a\},C\setminus\{a\})} y_e \\
&\le \sum_{c\in C\setminus\{a\}}\ \sum_{e\in\delta(c)\setminus\{(c,b)\}} y_e + \sum_{e\in E(\{a\},C\setminus\{a\})} y_e \\
&= \sum_{c\in C\setminus\{a\}}\ \sum_{e\in\delta(c)} y_e - \sum_{e\in E(\{b\},C\setminus\{a\})} y_e + \sum_{e\in E(\{a\},C\setminus\{a\})} y_e \\
&= \sum_{c\in C\setminus\{a\}}\ \sum_{e\in\delta(c)} y_e \\
&\le |C|-1.
\end{aligned}$$

□

Notons cependant que l'algorithme qui se déduit de cette construction n'est pas en général polynomial. Cependant, dans le cas particulier où $\sum_{v\in V(G)} b(v) = O(n)$, nous pouvons résoudre le PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM en un temps $O(n^3)$ (en utilisant l'ALGORITHME DE COUPLAGE AVEC POIDS ; voir corollaire 11.10). Pulleyblank [1973,1980] a donné une description des facettes de ce polytope et a montré que le système d'inégalités du théorème 12.2 est TDI. La généralisation suivante permet l'introduction de capacités :

Théorème 12.3. (Edmonds et Johnson [1970]) *Soit G un graphe orienté, $u : E(G) \to \mathbb{N}\cup\{\infty\}$ et $b : V(G) \to \mathbb{N}$. Le polytope du b-couplage de (G,u) est l'ensemble des vecteurs $x \in \mathbb{R}_+^{E(G)}$ qui vérifient*

$$\begin{aligned}
x_e &\le u(e) && (e\in E(G));\\
\sum_{e\in\delta(v)} x_e &\le b(v) && (v\in V(G));\\
\sum_{e\in E(G[X])} x_e + \sum_{e\in F} x_e &\le \left\lfloor \frac{1}{2}\left(\sum_{v\in X} b(v) + \sum_{e\in F} u(e)\right)\right\rfloor && (X\subseteq V(G),\ F\subseteq\delta(X)).
\end{aligned}$$

Preuve. Observons que le vecteur d'incidence f de tout b-couplage vérifie les contraintes. C'est évident sauf pour les dernières ; montrons donc cela : soit $X \subseteq V(G)$ et $F \subseteq \delta(X)$. Nous avons une offre de $b(v)$ unités en chaque sommet $v \in X$ et une offre de $u(e)$ unités sur chaque arête $e \in F$. Pour tout $e \in E(G[X])$ nous prélevons $f(e)$ unités de cette offre sur les deux extrémités de e. Pour tout $e \in F$, $e = (x,y)$ avec $x \in X$, nous prélevons $f(e)$ unités de cette offre en x et $f(e)$ unités sur l'arête e. On ne peut prélever plus que l'offre totale, et nous avons prélevé $2\sum_{e\in E(G[X])\cup F} f(e)$ unités. Donc

$$\sum_{e\in E(G[X])} x_e + \sum_{e\in F} x_e \le \frac{1}{2}\left(\sum_{v\in X} b(v) + \sum_{e\in F} u(e)\right).$$

Puisque le membre de gauche est un entier, le membre de droite l'est aussi ; on peut donc l'arrondir à sa valeur entière inférieure.

Soit alors $x \in \mathbb{R}_+^{E(G)}$ un vecteur tel que $x_e \leq u(e)$ pour tout $e \in E(G)$, $\sum_{e\in\delta(v)} x_e \leq b(v)$ pour tout $v \in V(G)$ et

$$\sum_{e\in E(G[X])} x_e + \sum_{e\in F} x_e \;\leq\; \left\lfloor \frac{1}{2}\left(\sum_{v\in X} b(v) + \sum_{e\in F} u(e)\right)\right\rfloor$$

pour tout $X \subseteq V(G)$ et $F \subseteq \delta(X)$. Montrons que x est combinaison convexe des vecteurs d'incidence des b-couplages de (G, u).

Soit H obtenu à partir de G par subdivision de chaque arête $e = (v, w)$ telle que $u(e) \neq \infty$ en ajoutant deux nouveaux sommets ev, ew. (À la place de e, H aura les arêtes (v, ev), (ev, ew) et (ew, w).) Soit $b(ev) := b(ew) := u(e)$ pour les nouveaux sommets.

Posons $y_{(v,ev)} := y_{(ew,w)} := x_e$ et $y_{(ev,ew)} := u(e) - x_e$ pour toute arête subdivisée $e = (v, w)$. Posons $y_e := x_e$ pour toute arête originale e avec $u(e) = \infty$. Nous affirmons que y appartient à P, polytope du b-couplage de (H, ∞).

Nous utiliserons le théorème 12.2. On a $y \in \mathbb{R}_+^{E(H)}$ et $\sum_{e\in\delta(v)} y_e \leq b(v)$ pour tout $v \in V(H)$. Supposons qu'il existe un ensemble $A \subseteq V(H)$ avec

$$\sum_{e\in E(H[A])} y_e \;>\; \left\lfloor \frac{1}{2}\sum_{a\in A} b(a)\right\rfloor. \tag{12.1}$$

Soit $B := A \cap V(G)$. Si $e = (v, w) \in E(G[B])$, nous pouvons supposer $ev, ew \in A$, car sinon l'addition de ev et ew à A induirait une autre inégalité (12.1). Nous pouvons également supposer que $ev \in A$ implique $v \in A$: si $ev, ew \in A$ mais $v \notin A$, nous pouvons retirer ev et ew de A sans détruire (12.1). Si $ev \in A$ et $v, ew \notin A$, nous pouvons retirer ev de A. La figure 12.1 montre les situations possibles pour les arêtes.

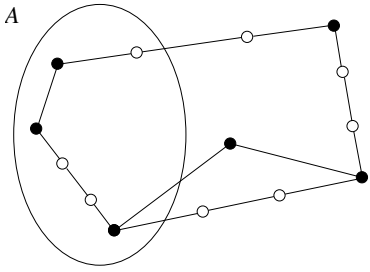


Figure 12.1.

Soit $F := \{e = (v, w) \in E(G) : |A \cap \{ev, ew\}| = 1\}$. Il vient

$$\begin{aligned}\sum_{e\in E(G[B])} x_e + \sum_{e\in F} x_e &= \sum_{e\in E(H[A])} y_e - \sum_{\substack{e\in E(G[B]),\\ u(e)<\infty}} u(e)\\ &> \left\lfloor \frac{1}{2}\sum_{a\in A} b(a)\right\rfloor - \sum_{\substack{e\in E(G[B]),\\ u(e)<\infty}} u(e)\\ &= \left\lfloor \frac{1}{2}\left(\sum_{v\in B} b(v) + \sum_{e\in F} u(e)\right)\right\rfloor,\end{aligned}$$

ce qui contredit notre hypothèse. Donc $y \in P$, et de plus y appartient à la face

$$\left\{ z \in P : \sum_{e\in\delta(v)} z_e = b(v) \text{ pour tout } v \in V(H)\setminus V(G)\right\}$$

de P. Puisque les sommets de cette face sont aussi sommets de P, y est combinaison convexe de b-couplages $f_1,\dots,f_m$ dans (H,∞), vérifiant $\sum_{e\in\delta(v)} f_i(e) = b(v)$ pour tout $v \in V(H)\setminus V(G)$. Cela implique $f_i((v,ev)) = f_i((ew,w)) \le u(e)$ pour chaque arête subdivisée $e = (v,w) \in E(G)$. Revenant à G, on voit que x est combinaison convexe des vecteurs d'incidence des b-couplages dans (G,u). □

Les constructions dans les preuves 12.2 et 12.3 ont été proposées par Tutte [1954]. Elles peuvent être également utilisées pour prouver une généralisation du théorème de Tutte 10.13 (exercice 4) :

Théorème 12.4. (Tutte [1952]) *Soit G un graphe, $u : E(G) \to \mathbb{N}\cup\{\infty\}$ et $b : V(G) \to \mathbb{N}$. (G,u) a un b-couplage parfait si et seulement si, pour toute paire de sous-ensembles disjoints $X, Y \subseteq V(G)$, le nombre de composantes connexes C de $G - X - Y$ avec $\sum_{c\in V(C)} b(c) + \sum_{e\in E_G(V(C),Y)} u(e)$ impair n'excède pas*

$$\sum_{v\in X} b(v) + \sum_{y\in Y}\left(\sum_{e\in\delta(y)} u(e) - b(y)\right) - \sum_{e\in E_G(X,Y)} u(e).$$

12.2 *T*-joints de poids minimum

Étudions le problème suivant : un postier doit distribuer le courrier dans sa zone. Il quitte donc le centre de tri, parcourt chaque rue au moins une fois, puis retourne au centre de tri. Le problème est de trouver une tournée de longueur minimum. Cela est connu sous le nom du PROBLÈME DU POSTIER CHINOIS (Guan [1962]).

Nous modéliserons les rues de la zone par un graphe supposé connexe. (Sinon le postier devra utiliser des rues hors de sa zone, et dans ce cas le problème est *NP*-complet ; voir l'exercice 14(d) du chapitre 15.) Le théorème d'Euler 2.24 dit qu'il

existe une tournée du postier qui utilise chaque arête une fois exactement (c.-à-d. un parcours eulérien) si et seulement si chaque sommet a un degré pair.

Si le graphe n'est pas eulérien, il nous faut utiliser certaines arêtes plusieurs fois. À partir du théorème d'Euler, nous pouvons ainsi formuler le PROBLÈME DU POSTIER CHINOIS : soit un graphe G et des poids $c : E(G) \to \mathbb{R}_+$; trouver une fonction $n : E(G) \to \mathbb{N}$ telle que le graphe G' obtenu à partir de G en ajoutant $n(e)$ copies de chaque arête $e \in E(G)$ soit eulérien avec $\sum_{e \in E(G)} n(e)c(e)$ minimum.

Cela est vrai que le graphe soit orienté ou non orienté. Dans le cas orienté, on peut utiliser des techniques de flots (exercice 9 du chapitre 9). Nous ne nous intéresserons donc désormais qu'aux graphes non orientés et nous aurons besoin dans ce cas de l'ALGORITHME DU COUPLAGE AVEC POIDS.

Il est clair qu'il n'est jamais nécessaire de passer plus de deux fois par une arête e, car sinon, en retranchant 2 de $n(e)$, on obtiendrait une solution qui ne peut être plus mauvaise. Le problème est donc de trouver un ensemble $J \subseteq E(G)$ de poids minimum tel que $(V(G), E(G) \dot\cup J)$ (le graphe obtenu en doublant les arêtes de J) soit eulérien. Nous allons résoudre dans ce paragraphe un problème plus général que le précédent.

Définition 12.5. *Soit G un graphe non orienté et $T \subseteq V(G)$ de cardinalité paire. $J \subseteq E(G)$ est un* ***T*-joint** *si $|J \cap \delta(x)|$ est impair si et seulement si $x \in T$.*

PROBLÈME DU T-JOINT DE POIDS MINIMUM	
Instance	Un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}$, et un ensemble $T \subseteq V(G)$ de cardinalité paire.
Tâche	Trouver un T-joint de poids minimum de G ou décider qu'il n'existe pas de T-joint.

Le PROBLÈME DU T-JOINT DE POIDS MINIMUM inclut de nombreux problèmes d'optimisation combinatoire :

- si c est non négatif et T est l'ensemble des sommets de degré impair de G, nous retrouvons le PROBLÈME DU POSTIER CHINOIS NON ORIENTÉ ;
- si $T = \emptyset$, les T-joints sont les sous-graphes eulériens. Donc l'ensemble vide est un $\emptyset$-joint de poids minimum si et seulement si c est conservatif ;
- si $|T| = 2$ et $T = \{s, t\}$, tout T-joint est l'union d'une chaîne de s à t et éventuellement de cycles. Donc, si c est conservatif, le PROBLÈME DU T-JOINT DE POIDS MINIMUM est équivalent au PROBLÈME DE LA PLUS COURTE CHAÎNE (notons que nous n'avons pas été capables de résoudre le PROBLÈME DE LA PLUS COURTE CHAÎNE dans les graphes non orientés au chapitre 7, sauf quand les coûts sont non négatifs) ;
- si $T = V(G)$, les T-joints de cardinalité $\frac{|V(G)|}{2}$ sont les couplages parfaits. Donc le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM se réduit au PROBLÈME DU T-JOINT DE POIDS MINIMUM en ajoutant une grande constante au poids de chaque arête.

L'objet de ce paragraphe est de proposer un algorithme polynomial pour le PROBLÈME DU T-JOINT DE POIDS MINIMUM. La question de l'existence d'un T-joint se résoudra simplement :

Proposition 12.6. *Soit G un graphe et $T \subseteq V(G)$ avec $|T|$ pair. Il existe un T-joint dans G si et seulement si $|V(C) \cap T|$ est pair pour chaque composante connexe C de G.*

Preuve. Si J est un T-joint $\sum_{v \in V(C)} |J \cap \delta(v)| = 2|J \cap E(C)|$ pour toute composante connexe C de G et $|J \cap \delta(v)|$ est impair pour un nombre pair de sommets $v \in V(C)$. Puisque J est un T-joint, $|V(C) \cap T|$ est pair.

Inversement, si $|V(C) \cap T|$ est pair pour chaque composante connexe C de G, T peut être partitionné en paires $\{v_1, w_1\}, \ldots, \{v_k, w_k\}$, avec $k = \frac{|T|}{2}$, de telle sorte que v_i et w_i soient dans la même composante connexe pour $i = 1, \ldots, k$. Soit P_i une chaîne de v_i à w_i ($i = 1, \ldots, k$), et soit $J := E(P_1) \triangle E(P_2) \triangle \cdots \triangle E(P_k)$. Puisque le degré de chaque sommet a la même parité à l'égard des ensembles J et $E(P_1) \dot\cup E(P_2) \dot\cup \cdots \dot\cup E(P_k)$, J est bien un T-joint. □

Un critère simple d'optimalité est :

Proposition 12.7. *Un T-joint J dans un graphe G avec poids $c : E(G) \to \mathbb{R}$ est de coût minimum si et seulement si $c(J \cap E(C)) \leq c(E(C) \setminus J)$ pour tout cycle C dans G.*

Preuve. Si $c(J \cap E(C)) > c(E(C) \setminus J)$, alors $J \triangle E(C)$ est un T-joint de poids plus petit que le poids de J. D'autre part, si J' est un T-joint avec $c(J') < c(J)$, $J' \triangle J$ est eulérien, c.-à-d. est une union de cycles, et on a pour un de ces cycles C $c(J \cap E(C)) > c(J' \cap E(C)) = c(E(C) \setminus J)$. □

On peut voir cette proposition comme un cas particulier du théorème 9.6. Nous allons maintenant résoudre le PROBLÈME DU T-JOINT DE POIDS MINIMUM avec des poids non négatifs en le réduisant au PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM. L'idée principale est contenue dans le lemme suivant :

Lemme 12.8. *Soit G un graphe, $c : E(G) \to \mathbb{R}_+$, et $T \subseteq V(G)$ avec $|T|$ pair. Tout T-joint optimum est une union disjointe d'ensembles d'arêtes de $\frac{|T|}{2}$ chaînes ayant des extrémités distinctes dans T, et éventuellement de cycles de poids nul.*

Preuve. Par induction sur $|T|$. Le cas $T = \emptyset$ est trivial puisque le poids d'un $\emptyset$-joint est zéro.

Soit J un T-joint optimum dans G ; nous pouvons toujours supposer que J ne contient pas de cycles de poids nul. Par la proposition 12.7, J ne contient pas de cycles de poids positif. Comme c est non négatif, J est une forêt. Soient x, y deux feuilles de la même composante connexe ($|J \cap \delta(x)| = |J \cap \delta(y)| = 1$) et soit P la chaîne de x à y dans J. $x, y \in T$ et $J \setminus E(P)$ est un $(T \setminus \{x, y\})$-joint de poids minimum (un $(T \setminus \{x, y\})$-joint J' de poids plus petit impliquerait que le T-joint $J' \triangle E(P)$ a un poids plus petit que celui de J). La condition de l'énoncé se déduit de notre hypothèse d'induction. □

Théorème 12.9. (Edmonds et Johnson [1973]) *Quand les poids sont non négatifs, la complexité du* PROBLÈME DU T-JOINT DE POIDS MINIMUM *est* $O(n^3)$.

Preuve. Soit (G, c, T) une instance. Nous résolvons d'abord le PROBLÈME DE TOUTES LES PLUS COURTES CHAÎNES dans (G, c) ; plus précisément, dans le graphe où chaque arête est remplacée par deux arcs de direction opposée de même poids. Par le théorème 7.9 cela nécessite un temps $O(mn+n^2 \log n)$. Nous obtenons en particulier la fermeture métrique $(\bar{G}, \bar{c})$ de (G, c) (voir corollaire 7.11).

Nous cherchons alors un couplage parfait de poids minimum M dans $(\bar{G}[T], \bar{c})$. Par le corollaire 11.10, cela nécessite un temps $O(n^3)$. Par le lemme 12.8, $\bar{c}(M)$ est au plus le poids minimum d'un T-joint.

Considérons la plus courte chaîne de x à y dans G pour chaque $(x, y) \in M$ (que nous avons déjà calculé). Soit J la différence symétrique des ensembles d'arêtes de ces chaînes. J est clairement un T-joint dans G. De plus, $c(J) \leq \bar{c}(M)$, donc J est optimum. □

Cette méthode ne peut plus s'appliquer quand certains coûts sont négatifs, puisqu'on pourrait introduire des cycles négatifs. Cependant, nous pouvons réduire le PROBLÈME DU T-JOINT DE POIDS MINIMUM avec des poids arbitraires au cas où tous les poids sont non négatifs :

Théorème 12.10. *Soit G un graphe avec des poids $c : E(G) \to \mathbb{R}$, et soit $T \subseteq V(G)$ un ensemble de sommets de cardinalité paire. Soit E^- l'ensemble des arêtes ayant un coût négatif et soit T^- l'ensemble des sommets incidents à un nombre impair d'arêtes négatives, et soit $d : E(G) \to \mathbb{R}_+$ avec $d(e) := |c(e)|$.*

Alors J est un T-joint de poids minimum avec les poids c si et seulement si $J \triangle E^-$ est un $(T \triangle T^-)$-joint de poids minimum avec les poids d.

Preuve. Pour tout sous-ensemble J de $E(G)$ nous avons

$$\begin{aligned} c(J) &= c(J \setminus E^-) + c(J \cap E^-) \\ &= c(J \setminus E^-) + c(J \cap E^-) + c(E^- \setminus J) + d(E^- \setminus J) \\ &= d(J \setminus E^-) + c(J \cap E^-) + c(E^- \setminus J) + d(E^- \setminus J) \\ &= d(J \triangle E^-) + c(E^-) \ . \end{aligned}$$

Alors J est un T-joint si et seulement si $J \triangle E^-$ est un $(T \triangle T^-)$-joint, ce qui prouve le théorème grâce à l'égalité précédente (puisque $c(E^-)$ est constant). □

Corollaire 12.11. *Le* PROBLÈME DU T-JOINT DE POIDS MINIMUM *peut se résoudre en un temps $O(n^3)$.*

Preuve. Conséquence directe des théorèmes 12.9 et 12.10. □

En utilisant l'implémentation la plus rapide pour le PROBLÈME DU COUPLAGE DE POIDS MAXIMUM, on peut trouver un T-joint en un temps $O(nm + n^2 \log n)$.

Finalement, nous pouvons résoudre le PROBLÈME DE LA PLUS COURTE CHAÎNE dans les graphes non orientés :

Corollaire 12.12. *La recherche d'une plus courte chaîne entre deux sommets donnés dans un graphe non orienté, avec des poids conservatifs, peut se résoudre en un temps $O(n^3)$.*

Preuve. Soient s et t deux sommets donnés. Soit $T := \{s,t\}$ et appliquons le théorème 12.11. Après avoir enlevé les cycles de poids nul, le T-joint obtenu est une chaîne de s à t. □

Cela implique également l'existence d'un algorithme en $O(mn^3)$ pour détecter un cycle de poids minimum dans un graphe non orienté avec des poids conservatifs (et en particulier le calcul de la maille). Pour résoudre le PROBLÈME DE LA PLUS COURTE CHAÎNE ENTRE TOUTES LES PAIRES dans les graphes non orientés, il faut effectuer $\binom{n}{2}$ calculs de couplage avec poids (ce qui requiert un temps $O(n^5)$). En utilisant les résultats de postoptimalité du paragraphe 11.4 nous pouvons montrer :

Théorème 12.13. *Le problème de la recherche de la plus courte chaîne entre toutes les paires de sommets d'un graphe non orienté G ayant des coûts conservatifs $c : E(G) \to \mathbb{R}$ peut se résoudre en un temps $O(n^4)$.*

Preuve. Par le théorème 12.10 et la preuve du corollaire 12.12 il faut calculer un $(\{s,t\}\triangle T^-)$-joint optimum par rapport aux poids $d(e) := |c(e)|$ pour tous $s,t \in V(G)$, T^- étant l'ensemble des sommets incidents à un nombre impair d'arêtes négatives. Soit $\bar{d}(\{x,y\}) := \mathrm{dist}_{(G,d)}(x,y)$ pour $x,y \in V(G)$, et soit H_X le graphe complet sur $X\triangle T^-$ ($X \subseteq V(G)$). Il est suffisant, par la preuve du théorème 12.9, de calculer un couplage parfait de poids minimum dans $\left(H_{\{s,t\}}, \bar{d}\right)$ pour tous les s et t.

Notre algorithme en $O(n^4)$ se déroule ainsi : nous calculons d'abord $\bar{d}$ (voir corollaire 7.11) et exécutons l'ALGORITHME DU COUPLAGE AVEC POIDS sur l'instance $(H_\emptyset, \bar{d})$. Le temps de calcul est alors $O(n^3)$.

Montrons que nous pouvons calculer ensuite un couplage parfait de poids minimum dans $\left(H_{\{s,t\}}, \bar{d}\right)$ en un temps $O(n^2)$, quels que soient s et t.

Soit $K := \sum_{e\in E(G)} \bar{d}(e)$, et $s,t \in V(G)$. Il y aura quatre cas :

Cas 1 : $s,t \in T^-$. Attribuons à l'arête (s,t) un poids $-K$. Après une nouvelle optimisation (utilisant le lemme 11.11), (s,t) doit appartenir au couplage optimum M, et $M \setminus \{(s,t)\}$ est un couplage parfait de poids minimum de $\left(H_{\{s,t\}}, \bar{d}\right)$.

Cas 2 : $s \in T^-$ et $t \notin T^-$. Attribuons à l'arête (s,v) le poids $\bar{d}((t,v))$ quel que soit $v \in T^- \setminus \{s\}$. s joue le rôle de t, et en optimisant de nouveau (en utilisant le lemme 11.11) nous obtenons le résultat.

Cas 3 : $s \notin T^-$ et $t \in T^-$. Cas symétrique du cas 2.

Cas 4 : $s,t \notin T^-$. Ajoutons s et t et appliquons le lemme 11.12. □

12.3 *T*-joints et *T*-coupes

Nous allons maintenant associer une description polyédrale au PROBLÈME DU T-JOINT DE POIDS MINIMUM. Dans la description du polytope du couplage parfait

(théorème 11.13), une contrainte de coupe $\delta(X)$ était associée à tout X avec $|X|$ impair ; ici nous aurons des contraintes associées à chaque T-coupe. Une ***T*-coupe** est une coupe $\delta(X)$ avec $|X \cap T|$ impair. L'observation suivante sera utile :

Proposition 12.14. *Soit G un graphe non orienté et soit $T \subseteq V(G)$ avec $|T|$ pair. Alors pour tout T-joint J et toute T-coupe C, $J \cap C \neq \emptyset$.*

Preuve. Supposons que $C = \delta(X)$; alors $|X \cap T|$ est impair. Donc le nombre d'arêtes dans $J \cap C$ est impair, en particulier différent de 0. □

On trouvera une condition plus forte dans l'exercice 11.

La proposition 12.14 implique que la cardinalité minimum d'un T-joint est supérieure ou égale au nombre maximum de T-coupes arête-disjointes. En général nous n'avons pas l'égalité : soit par exemple $G = K_4$ avec $T = V(G)$. Cependant dans les graphes bipartis, nous avons l'égalité :

Théorème 12.15. (Seymour [1981]) *Soit G un graphe biparti connexe et soit $T \subseteq V(G)$ avec $|T|$ pair : la cardinalité minimum d'un T-joint est égale au nombre maximum de T-coupes arête-disjointes.*

Preuve. (Sebő [1987]) Nous devons seulement prouver l'inégalité $\leq$. Nous ferons une induction sur $|V(G)|$. Si $T = \emptyset$ (en particulier si $|V(G)| = 1$), le résultat est trivial. Supposons donc que $|V(G)| \geq |T| \geq 2$. Soit $\tau(G,T)$ la cardinalité minimum d'un T-joint de G. Choisissons $a, b \in V(G)$, $a \neq b$, de telle sorte que $\tau(G, T\triangle\{a,b\})$ soit minimum et soit $T' := T\triangle\{a,b\}$. Puisque nous pouvons supposer que $T \neq \emptyset$, $\tau(G,T') < \tau(G,T)$.

Affirmation : si J est un T-joint minimum de G $|J \cap \delta(a)| = |J \cap \delta(b)| = 1$.

Montrons cela ; soit J' un T'-joint minimum. $J\triangle J'$ est l'union arête-disjointe d'une chaîne de a à b P et de cycles $C_1, \ldots, C_k$. $|C_i \cap J| = |C_i \cap J'|$ pour chaque i, car J et J' sont tous deux minimum. Donc $|J\triangle P| = |J'|$, et $J'' := J\triangle P$ est aussi un T'-joint minimum. $J'' \cap \delta(a) = J'' \cap \delta(b) = \emptyset$, parce que si $(b,b') \in J''$ pour un sommet b' $J'' \setminus \{\{b,b'\}\}$ est un $(T\triangle\{a\}\triangle\{b'\})$-joint, et $\tau(G, T\triangle\{a\}\triangle\{b'\}) < |J''| = |J'| = \tau(G,T')$, ce qui contredit notre choix de a et b. En conclusion $|J \cap \delta(a)| = |J \cap \delta(b)| = 1$ et cela termine la preuve de l'affirmation.

En particulier, $a, b \in T$. Soit alors J un T-joint minimum dans G. Contractons $B := \{b\} \cup \Gamma(b)$ en un seul sommet v_B et soit G^* le graphe résultant. G^* est aussi biparti. Soit $T^* := T \setminus B$ si $|T \cap B|$ est pair et $T^* := (T \setminus B) \cup \{v_B\}$ dans le cas contraire. L'ensemble J^*, résultant de J par la contraction de B, est manifestement un T^*-joint de G^*. Puisque $\Gamma(b)$ est un ensemble stable de G (G étant biparti), l'affirmation implique que $|J| = |J^*| + 1$.

Il suffit de montrer que J^* est un T^*-joint minimum de G^*, puisque alors $\tau(G,T) = |J| = |J^*| + 1 = \tau(G^*, T^*) + 1$, et le théorème sera démontré par induction (observons que $\delta(b)$ est une T-coupe de G disjointe de $E(G^*)$).

Supposons donc que J^* ne soit pas un T^*-joint minimum de G^*. Par la proposition 12.7, il existe un cycle C^* dans G^* avec $|J^* \cap E(C^*)| > |E(C^*) \setminus J^*|$. Puisque G^* est biparti, $|J^* \cap E(C^*)| \geq |E(C^*) \setminus J^*| + 2$. $E(C^*)$ correspond à un

ensemble d'arêtes Q de G. Q ne peut être un cycle parce que $|J \cap Q| > |Q \setminus J|$ et J est un T-joint minimum. Donc Q est une chaîne de x à y dans G pour une paire $x, y \in \Gamma(b)$ telle que $x \neq y$. Soit C un cycle dans G formé par Q, (x, b) et (b, y). Puisque J est un T-joint minimum dans G,

$$|J \cap E(C)| \leq |E(C) \setminus J| \leq |E(C^*) \setminus J^*| + 2 \leq |J^* \cap E(C^*)| \leq |J \cap E(C)|.$$

Donc les inégalités sont des égalités et en particulier $(x, b), (b, y) \notin J$ et $|J \cap E(C)| = |E(C) \setminus J|$. Donc $\bar{J} := J \triangle E(C)$ est un T-joint minimum et $|\bar{J} \cap \delta(b)| = 3$. ce qui est impossible par l'affirmation. □

Les T-coupes sont essentielles dans la description du polyèdre des T-joints :

Théorème 12.16. (Edmonds et Johnson [1973]) *Soit G un graphe non orienté, $c : E(G) \to \mathbb{R}_+$, et $T \subseteq V(G)$ avec $|T|$ pair. Alors le vecteur d'incidence d'un T-joint de poids minimum est solution optimale du PL*

$$\min \left\{ cx : x \geq 0, \sum_{e \in C} x_e \geq 1 \textit{ pour toute } T\textit{-coupe } C \right\}.$$

(Ce polyèdre est appelé le **polyèdre des *T*-joints** *de G.)*

Preuve. Par la proposition 12.14, le vecteur d'incidence d'un T-joint vérifie les contraintes. Soit $c : E(G) \to \mathbb{R}_+$ un ensemble donné de poids ; nous pouvons supposer que $c(e)$ est un entier pair pour tout $e \in E(G)$. Soit k le poids minimum d'un T-joint de G (par rapport à c). Montrons que k est la valeur du PL de l'énoncé.

Remplaçons chaque arête e par une chaîne de longueur $c(e)$ (si $c(e) = 0$, contractons e et ajoutons le sommet contracté à T si et seulement si $|e \cap T| = 1$). Le graphe résultant G' est biparti. De plus, la cardinalité minimum d'un T-joint de G' est k. Par le théorème 12.15, il existe une famille $\mathcal{C}'$ de k T-coupes arête-disjointes dans G'. Cela fournit une famille $\mathcal{C}$ de k T-coupes dans G et chaque arête e est contenue dans au plus $c(e)$ de ces coupes. Donc, si x est une solution réalisable du PL, nous avons

$$cx \geq \sum_{C \in \mathcal{C}} \sum_{e \in C} x_e \geq \sum_{C \in \mathcal{C}} 1 = k,$$

montrant ainsi que la valeur optimale est k. □

Cela implique le théorème 11.13 : supposons que G ait un couplage parfait et posons $T := V(G)$. Alors le théorème 12.16 implique que

$$\min \left\{ cx : x \geq 0, \sum_{e \in C} x_e \geq 1 \text{ pour toute } T\text{-coupe } C \right\}$$

est entier pour tout $c \in \mathbb{Z}^{E(G)}$ pour lequel le minimum est fini. Par le théorème 5.13, le polyèdre est entier, ainsi que sa face

$$\left\{ x \in \mathbb{R}_+^{E(G)} : \sum_{e \in C} x_e \geq 1 \text{ pour toute } T\text{-coupe } C, \sum_{e \in \delta(v)} x_e = 1 \text{ pour tout } v \in V(G) \right\}.$$

Il est possible de déduire une description équivalente de ce polyèdre (exercice 14). Les théorèmes 12.16 et 4.21 (ainsi que le corollaire 3.33) impliquent l'existence d'un algorithme polynomial pour le PROBLÈME DU T-JOINT DE POIDS MINIMUM pourvu que nous sachions résoudre le PROBLÈME DE SÉPARATION dans la description ci-dessus. Cela est clairement équivalent au problème de l'existence d'une T-coupe de capacité plus petite que 1 (ici x sera un vecteur capacités). Il suffit donc de résoudre le problème suivant :

PROBLÈME DE LA T-COUPE DE CAPACITÉ MINIMUM	
Instance	Un graphe G, des capacités $u : E(G) \to \mathbb{R}_+$, et un ensemble $T \subseteq V(G)$ de cardinalité paire.
Tâche	Trouver une T-coupe de G de capacité minimum.

Notons que le PROBLÈME DE LA T-COUPE DE CAPACITÉ MINIMUM résout le PROBLÈME DE SÉPARATION pour le polytope des couplages parfaits (théorème 11.13 ; $T := V(G)$). Le théorème suivant résout également le PROBLÈME DE LA T-COUPE DE CAPACITÉ MINIMUM : il suffit d'étudier les coupes fondamentales de l'arbre de Gomory-Hu. Rappelons que nous savons trouver un arbre de Gomory-Hu dans un graphe non orienté avec capacités en un temps $O(n^4)$ (théorème 8.35).

Théorème 12.17. (Padberg et Rao [1982]) *Soit G un graphe non orienté avec des capacités $u : E(G) \to \mathbb{R}_+$. Soit H un arbre de Gomory-Hu pour (G, u). Soit $T \subseteq V(G)$ avec $|T|$ pair. Il existe une T-coupe de capacité minimum parmi les coupes fondamentales de H. On peut donc trouver la capacité minimum d'une T-coupe en un temps $O(n^4)$.*

Preuve. Considérons la paire $(G + H, u')$ avec $u'(e) = u(e)$ pour $e \in E(G)$ et $u'(e) = 0$ pour $e \in E(H)$. Soit $A \subseteq E(G) \cup E(H)$ une T-coupe dans $(G + H, u')$. Il est évident que $u'(A) = u(A \cap E(G))$ et $A \cap E(G)$ est une T-coupe dans (G, u).

Soit J l'ensemble des arêtes e de H tel que $\delta_G(C_e)$ soit une T-coupe. On voit facilement que J est un T-joint (dans $G + H$). Par la proposition 12.14, il existe une arête $e = (v, w) \in A \cap J$. Nous avons alors

$$u(A \cap E(G)) \;\geq\; \lambda_{vw} \;=\; \sum_{(x,y) \in \delta_G(C_e)} u((x, y)),$$

ce qui montre que $\delta_G(C_e)$ est une T-coupe minimum. □

12.4 Théorème de Padberg-Rao

La solution du PROBLÈME DE LA T-COUPE DE CAPACITÉ MINIMUM nous aide également à résoudre le PROBLÈME DE SÉPARATION pour le polytope du b-couplage (théorème 12.3) :

Théorème 12.18. (Padberg et Rao [1982]) *Si G est un graphe non orienté avec $u : E(G) \to \mathbb{N} \cup \{\infty\}$ et $b : V(G) \to \mathbb{N}$, le* PROBLÈME DE SÉPARATION *pour le polytope du b-couplage de (G, u) peut se résoudre en temps polynomial.*

Preuve. Nous pouvons supposer $u(e) < \infty$ pour toutes les arêtes e (on peut remplacer les capacités infinies par un nombre assez grand, par exemple $\max\{b(v) : v \in V(G)\}$). Choisissons pour G une orientation fixée ; nous utiliserons quelquefois les arêtes de G et d'autres fois les arcs du graphe orienté.

Soit $x \in \mathbb{R}_+^{E(G)}$ tel que $x_e \leq u(e)$ pour tout $e \in E(G)$ et $\sum_{e \in \delta_G(v)} x_e \leq b(v)$ pour tout $v \in V(G)$ (ces inégalités triviales peuvent être vérifiées en temps linéaire) ; définissons un nouveau graphe biparti H avec des capacités $t : E(H) \to \mathbb{R}_+$ de la manière suivante :

$$\begin{aligned}
V(H) &:= V(G) \,\dot\cup\, E(G) \,\dot\cup\, \{S\},\\
E(H) &:= \{(v,e) : v \in e \in E(G)\} \cup \{(v,S) : v \in V(G)\},\\
t((v,e)) &:= u(e) - x_e && (e \in E(G), \text{ quand } v \text{ est l'origine de } e),\\
t((v,e)) &:= x_e && (e \in E(G), \text{ quand } v \text{ est l'extrémité de } e),\\
t((v,S)) &:= b(v) - \sum_{e \in \delta_G(v)} x_e && (v \in V(G)).
\end{aligned}$$

Soit $T \subseteq V(H)$ constitué :

- des sommets $v \in V(G)$ pour lesquels $b(v) + \sum_{e \in \delta_G^+(v)} u(e)$ est impair,
- des sommets $e \in E(G)$ pour lesquels $u(e)$ est impair, et
- du sommet S si $\sum_{v \in V(G)} b(v)$ est impair.

Notons que $|T|$ est pair.

Nous allons montrer qu'il existe une T-coupe dans H de capacité inférieure à un si et seulement si x n'est pas dans l'enveloppe des b-couplages de (G, u).

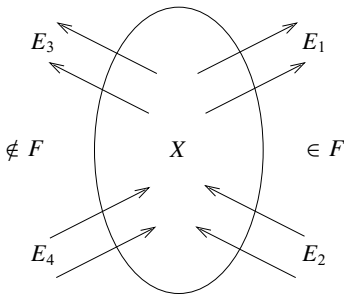


Figure 12.2.

Nous avons besoin de quelques résultats préalables. Soit $X \subseteq V(G)$ et soit $F \subseteq \delta_G(X)$. Posons

$$\begin{aligned} E_1 &:= \{e \in \delta_G^+(X) \cap F\}, \\ E_2 &:= \{e \in \delta_G^-(X) \cap F\}, \\ E_3 &:= \{e \in \delta_G^+(X) \setminus F\}, \\ E_4 &:= \{e \in \delta_G^-(X) \setminus F\}, \end{aligned}$$

(voir figure 12.2) et

$$W := X \cup E(G[X]) \cup E_2 \cup E_3 \subseteq V(H).$$

Affirmation :

(a) $|W \cap T|$ est impair si et seulement si $\sum_{v \in X} b(v) + \sum_{e \in F} u(e)$ est impair.

(b) $\sum_{e \in \delta_H(W)} t(e) < 1$ si et seulement si

$$\sum_{e \in E(G[X])} x_e + \sum_{e \in F} x_e > \frac{1}{2} \left(\sum_{v \in X} b(v) + \sum_{e \in F} u(e) - 1 \right).$$

Pour démontrer (a), il suffit d'observer que, par définition, $|W \cap T|$ est impair si et seulement si

$$\sum_{v \in X} \left(b(v) + \sum_{e \in \delta_G^+(v)} u(e) \right) + \sum_{e \in E(G[X]) \cup E_2 \cup E_3} u(e)$$

est impair. Mais ce nombre est égal à

$$\begin{aligned} & \sum_{v \in X} b(v) + 2 \sum_{e \in E(G[X])} u(e) + \sum_{e \in \delta_G^+(X)} u(e) + \sum_{e \in E_2 \cup E_3} u(e) \\ = & \sum_{v \in X} b(v) + 2 \sum_{e \in E(G[X])} u(e) + 2 \sum_{e \in \delta_G^+(X)} u(e) - 2 \sum_{e \in E_1} u(e) + \sum_{e \in E_1 \cup E_2} u(e), \end{aligned}$$

ce qui montre (a), puisque $E_1 \cup E_2 = F$. De plus,

$$\begin{aligned} \sum_{e \in \delta_H(W)} t(e) &= \sum_{\substack{e \in E_1 \cup E_4 \\ x \in e \cap X}} t(\{x, e\}) + \sum_{\substack{e \in E_2 \cup E_3 \\ y \in e \setminus X}} t(\{y, e\}) + \sum_{x \in X} t(\{x, S\}) \\ &= \sum_{e \in E_1 \cup E_2} (u(e) - x_e) + \sum_{e \in E_3 \cup E_4} x_e + \sum_{v \in X} \left(b(v) - \sum_{e \in \delta_G(v)} x_e \right) \\ &= \sum_{e \in F} u(e) + \sum_{v \in X} b(v) - 2 \sum_{e \in F} x_e - 2 \sum_{e \in E(G[X])} x_e, \end{aligned}$$

ce qui montre (b).

Prouvons qu'il existe une T-coupe dans H de capacité plus petite que un si et seulement si x n'est pas dans l'enveloppe convexe des b-couplages de (G, u). Supposons d'abord qu'il existe $X \subseteq V(G)$ et $F \subseteq \delta_G(X)$ avec

$$\sum_{e\in E(G[X])} x_e + \sum_{e\in F} x_e \;>\; \left\lfloor \frac{1}{2}\left(\sum_{v\in X} b(v) + \sum_{e\in F} u(e)\right)\right\rfloor .$$

Alors $\sum_{v\in X} b(v) + \sum_{e\in F} u(e)$ est certainement impair et

$$\sum_{e\in E(G[X])} x_e + \sum_{e\in F} x_e \;>\; \frac{1}{2}\left(\sum_{v\in X} b(v) + \sum_{e\in F} u(e) - 1\right).$$

Par (a) et (b), $\delta_H(W)$ est une T-coupe de capacité inférieure à un.

Pour montrer la réciproque, soit $\delta_H(W)$ une T-coupe quelconque dans H de capacité inférieure à un. Nous allons montrer comment construire une inégalité violée du polytope du b-couplage.

Nous pouvons toujours supposer $S \notin W$ (sinon échangeons W et $V(H)\setminus W$). Soit $X := W \cap V(G)$. Observons que $(v,(v,w)) \in \delta_H(W)$ implique $(v,w) \in \delta_G(X)$: Si $(v,w) \notin W$ pour une paire $v,w \in X$, les deux arcs $(v,(v,w))$ et $(w,(v,w))$ (avec capacité totale $u((v,w))$) appartiendraient à $\delta_H(W)$, infirmant l'hypothèse que cette coupe est de capacité inférieure à un. L'hypothèse $(v,w) \in W$ pour une paire $v,w \notin X$ conduit à la même conclusion. Soit

$$F \;:=\; \{(v,w) \in E(G) : (v,(v,w)) \in \delta_H(W)\}.$$

Par l'observation qui précède nous avons $F \subseteq \delta_G(X)$. En reprenant la définition de E_1, E_2, E_3, E_4 nous allons montrer que

$$W \;=\; X \cup E(G[X]) \cup E_2 \cup E_3 \tag{12.2}$$

est vérifié. De nouveau par l'observation précédente, nous avons seulement à montrer que $W \cap \delta_G(X) = E_2 \cup E_3$. Mais $e = (v,w) \in E_1 = \delta_G^+(X) \cap F$ implique $e \notin W$ par la définition de F. De même $e = (v,w) \in E_2 = \delta_G^-(X) \cap F$ implique $e \in W$, $e = (v,w) \in E_3 = \delta_G^+(X) \setminus F$ implique $e \in W$, et $e = (v,w) \in E_4 = \delta_G^-(X) \setminus F$ implique $e \notin W$. Donc (12.2) est démontré.

(a) et (b) sont ainsi vérifiés. Puisque $|W \cap T|$ est impair, (a) implique que $\sum_{v\in X} b(v) + \sum_{e\in F} u(e)$ est impair. Par (b) et l'hypothèse $\sum_{e\in\delta_H(W)} t(e) < 1$,

$$\sum_{e\in E(G[X])} x_e + \sum_{e\in F} x_e \;>\; \left\lfloor \frac{1}{2}\left(\sum_{v\in X} b(v) + \sum_{e\in F} u(e)\right)\right\rfloor ,$$

qui est une inégalité violée du polytope du b-couplage.

Nous avons montré que la capacité minimum d'une T-coupe de H est inférieure à un si et seulement si x viole une inégalité du polytope du b-couplage. De plus, étant donné une T-coupe dans H de capacité inférieure à un, nous pouvons facilement construire une inégalité violée. Donc le problème se réduit au PROBLÈME DE LA T-COUPE DE CAPACITÉ MINIMUM avec des coûts non négatifs. Par le théorème 12.17, ce problème se résout en un temps $O(n^4)$, ($n = |V(H)|$). □

On peut trouver une généralisation de ce résultat dans Caprara et Fischetti [1996]. Letchford, Reinelt et Theis [2004] ont montré qu'il suffisait de regarder l'arbre de Gomory-Hu pour (G, u). Ils réduisent le PROBLÈME DE SÉPARATION pour les inégalités du b-couplage (et d'autres plus générales) à un calcul de $|V(G)|$ flots maximum sur le graphe original, ce qui donne un algorithme en un $O(|V(G)|^4)$.

Le théorème de Padberg-Rao implique :

Corollaire 12.19. *Le* PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM *peut se résoudre en temps polynomial.*

Preuve. Par le corollaire 3.33 il faut résoudre le PL formulé au théorème 12.3. Par le théorème 4.21, il suffit d'avoir un algorithme polynomial pour le PROBLÈME DE SÉPARATION. Un tel algorithme est fourni par le théorème 12.18. □

Marsh [1979] a étendu l'ALGORITHME DU COUPLAGE AVEC POIDS d'Edmonds au PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM. Cet algorithme combinatoire est bien entendu plus performant que la MÉTHODE DES ELLIPSOÏDES. Cependant le théorème 12.18 est intéressant dans d'autres contextes (voir par exemple paragraphe 21.4). Pour un algorithme fortement polynomial, voir Anstee [1987] ou Gerards [1995].

Exercices

1. Montrer qu'on peut trouver un 2-couplage simple parfait de poids minimum dans un graphe non orienté G en un temps $O(n^6)$.

∗ 2. Soit G un graphe non orienté et soient $b_1, b_2 : V(G) \to \mathbb{N}$. Caractériser l'enveloppe convexe des fonctions $f : E(G) \to \mathbb{Z}_+$ avec $b_1(v) \leq \sum_{e \in \delta(v)} f(e) \leq b_2(v)$.
Indication : si $X, Y \subseteq V(G)$ avec $X \cap Y = \emptyset$ étudier la contrainte

$$\sum_{e \in E(G[X])} f(e) \; - \sum_{e \in E(G[Y]) \cup E(Y,Z)} f(e) \; \leq \; \left\lfloor \frac{1}{2} \left(\sum_{x \in X} b_2(x) - \sum_{y \in Y} b_1(y) \right) \right\rfloor ,$$

où $Z := V(G) \setminus (X \cup Y)$. Utiliser le théorème 12.3.
(Schrijver [1983])

∗ 3. Peut-on étendre les résultats de l'exercice 2 si on introduit des capacités inférieures et supérieures sur les arêtes ?
Note : cela peut être vu comme une version non orientée du problème posé à l'exercice 3 du chapitre 9. Pour une généralisation de ces deux problèmes et aussi du PROBLÈME DU T-JOINT DE POIDS MINIMUM, voir les articles d'Edmonds et Johnson [1973], et Schrijver [1983]. Même dans ce cas une description du polytope qui est TDI est connue.

$*$ 4. Montrer le théorème 12.4.
Indication : pour la condition suffisante, utiliser le théorème de Tutte 10.13 et les constructions dans les preuves des théorèmes 12.2 et 12.3.

5. Le polytope du sous-graphe de degrés donnés d'un graphe G est défini comme étant l'enveloppe convexe des vecteurs $b \in \mathbb{Z}_+^{V(G)}$ tels que G admette un b-couplage parfait simple. Montrer que sa dimension est $|V(G)| - k$, où k est le nombre de composantes connexes de G qui sont des graphes bipartis.

$*$ 6. Soit un graphe non orienté ; une couverture des cycles impairs par des arêtes est définie comme étant un ensemble d'arêtes contenant au moins une arête de chaque cycle impair. Montrer comment trouver en temps polynomial une couverture par des arêtes des cycles impairs de poids minimum dans un graphe planaire, les poids des arêtes étant non négatifs. Peut-on résoudre ce problème quand les poids sont quelconques ?
Indication : étudier le PROBLÈME DU POSTIER CHINOIS NON ORIENTÉ dans le graphe planaire et utiliser le théorème 2.26 et le corollaire 2.45.

7. Intéressons-nous au PROBLÈME DE LA COUPE DE POIDS MAXIMUM dans les graphes planaires : soit un graphe planaire non orienté G muni de poids $c : E(G) \to \mathbb{R}_+$; nous recherchons une coupe de poids maximum. Peut-on résoudre ce problème en temps polynomial ?
Indication : utiliser l'exercice 6.
Note : ce problème pour les graphes généraux est *NP*-difficile ; voir le théorème 16.6.
(Hadlock [1975])

8. Soit un graphe G avec des poids $c : E(G) \to \mathbb{R}_+$ et soit un ensemble $T \subseteq V(G)$ avec $|T|$ pair. Construisons un nouveau graphe G' en posant

$$\begin{aligned} V(G') &:= \{ve : v \text{ est une extrémité de } e \in E(G)\} \cup \\ &\quad \{\bar{v} : v \in V(G),\ |\delta_G(v)| + |\{v\} \cap T| \text{ impair}\}, \\ E(G') &:= \{(ve, we) : e = (v, w) \in E(G)\} \cup \\ &\quad \{(ve, vf) : v \in V(G),\ e, f \in \delta_G(v),\ e \neq f\} \cup \\ &\quad \{(\bar{v}, ve) : v \in e \in E(G),\ \bar{v} \in V(G')\}, \end{aligned}$$

et posons $c'((ve, we)) := c(e)$ pour $e = (v, w) \in E(G)$ et $c'(e') = 0$ pour toutes les autres arêtes dans G'.
Montrer qu'un couplage parfait de poids minimum dans G' correspond à un T-joint de poids minimum dans G. Cette réduction est-elle préférable à celle donnée dans la preuve du théorème 12.9 ?

$*$ 9. Le problème suivant relie les b-couplages simples et les T-joints : soit G un graphe non orienté muni de poids $c : E(G) \to \mathbb{R}$, soit une partition de l'ensemble des sommets $V(G) = R \dot\cup S \dot\cup T$, et soit une fonction $b : R \to \mathbb{Z}_+$. Nous cherchons un sous-ensemble d'arêtes $J \subseteq E(G)$ tel que $J \cap \delta(v) = b(v)$ pour $v \in R$, $|J \cap \delta(v)|$ soit pair pour $v \in S$, et $|J \cap \delta(v)|$ soit impair pour $v \in T$. Montrer comment réduire ce problème à un PROBLÈME DE COUPLAGE

PARFAIT DE POIDS MINIMUM.
Indication : étudier les constructions du paragraphe 12.1 et de l'exercice 8.

10. Considérons le PROBLÈME DU CYCLE MOYEN MINIMUM : soit un graphe non orienté G muni de poids $c : E(G) \to \mathbb{R}$: trouver un cycle C de G dont le coût moyen $\frac{c(E(C))}{|E(C)|}$ est minimum.
    (a) Montrer que l'ALGORITHME DU CIRCUIT MOYEN MINIMUM décrit au paragraphe 7.3 ne peut pas s'appliquer au cas non orienté.
    $*$ (b) Proposer un algorithme fortement polynomial pour le PROBLÈME DU CYCLE MOYEN DE COÛT MINIMUM.
    Indication : utiliser l'exercice 9.
11. Soit G un graphe non orienté, soit $T \subseteq V(G)$ avec $|T|$ pair, et soit $F \subseteq E(G)$. Montrer que : F a une intersection non vide avec chaque T-joint si et seulement si F contient une T-coupe. Montrer que F a une intersection non vide avec chaque T-coupe si et seulement si F contient un T-joint.

$*$ 12. Soit G un graphe planaire 2-connexe et soit une représentation plane de G ; soit C le cycle bordant la face externe, et soit T un sous-ensemble de sommets de $V(C)$ de cardinalité paire. Montrer que la cardinalité minimum d'un T-joint est égale au nombre maximum de T-coupes arête-disjointes.
Indication : colorer les arêtes de C en rouge et bleu, de telle sorte que quand on décrit C, les couleurs changent quand on traverse un sommet de T. Étudier le graphe planaire dual, séparer le sommet représentant la face extérieure, en un sommet rouge et un sommet bleu et appliquer le théorème de Menger 8.9.

13. Montrer le théorème 12.16 en utilisant le théorème 11.13 et la construction de l'exercice 8.
(Edmonds et Johnson [1973])

14. Soit G un graphe non orienté et soit $T \subseteq V(G)$ avec $|T|$ pair. Montrer que l'enveloppe convexe des vecteurs d'incidence des T-joints dans G est l'ensemble des vecteurs $x \in [0,1]^{E(G)}$ qui vérifient

$$\sum_{e \in \delta_G(X) \setminus F} x_e + \sum_{e \in F} (1 - x_e) \;\geq\; 1$$

pour tout $X \subseteq V(G)$ et $F \subseteq \delta_G(X)$ avec $|X \cap T| + |F|$ impair.
Indication : utiliser les théorèmes 12.16 et 12.10.

15. Soit G un graphe non orienté et soit $T \subseteq V(G)$ avec $|T| = 2k$ pair. Montrer que la cardinalité minimum d'une T-coupe dans G est égale au maximum de $\min_{i=1}^{k} \lambda_{s_i,t_i}$ pour tous les jumelages $T = \{s_1, t_1, s_2, t_2, \ldots, s_k, t_k\}$. ($\lambda_{s,t}$ représente le nombre maximum de chaînes de s à t arête-disjointes.) Peut-on étendre à une version avec poids cette formule min-max ?
Indication : utiliser le théorème 12.17.
(Rizzi [2002])

16. Cet exercice propose un algorithme pour le PROBLÈME D'UNE T-COUPE DE CAPACITÉ MINIMUM sans utiliser les arbres de Gomory-Hu. L'algorithme est récursif et – étant donné G, u et T – procède de la manière suivante :

    1. Nous partons d'abord d'un ensemble $X \subseteq V(G)$ avec $T \cap X \neq \emptyset$ et $T \setminus X \neq \emptyset$, tel que $u(X) := \sum_{e \in \delta_G(X)} u(e)$ est minimum (voir l'exercice 22 du chapitre 8). Si $|T \cap X|$ est impair, nous avons terminé (retourner X).

    2. Sinon nous appliquons l'algorithme de manière récursive en partant d'abord de G, u et $T \cap X$, et ensuite à G, u et $T \setminus X$. Nous obtenons $Y \subseteq V(G)$ avec $|(T \cap X) \cap Y|$ impair et $u(Y)$ minimum et $Z \subseteq V(G)$ avec $|(T \setminus X) \cap Z|$ impair et $u(Z)$ minimum. On peut toujours supposer que $T \setminus X \not\subseteq Y$ et que $X \cap T \not\subseteq Z$ (sinon on remplacera Y par $V(G) \setminus Y$ et/ou Z par $V(G) \setminus Z$).

    3. Si $u(X \cap Y) < u(Z \setminus X)$, alors retourner $X \cap Y$, sinon retourner $Z \setminus X$.

    Montrer que cet algorithme est correct et que son temps de calcul est $O(n^5)$ ($n = |V(G)|$).

17. Montrer comment résoudre le PROBLÈME DU b-COUPLAGE DE POIDS MAXIMUM quand $b(v)$ est pair pour tout $v \in V(G)$ en un temps fortement polynomial.

    Indication : réduction à un PROBLÈME DE FLOT DE COÛT MINIMUM comme dans l'exercice 10 du chapitre 9.

Références

Littérature générale :

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. [1998] : Combinatorial Optimization. Wiley, New York 1998, Sections 5.4 and 5.5

Frank, A. [1996] : A survey on T-joins, T-cuts, and conservative weightings. In : Combinatorics, Paul Erdős is Eighty ; Volume 2 (D. Miklós, V.T. Sós, T. Szőnyi, eds.), Bolyai Society, Budapest 1996, pp. 213–252

Gerards, A.M.H. [1995] : Matching. In : Handbooks in Operations Research and Management Science ; Volume 7 : Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995, pp. 135–224

Lovász, L., Plummer, M.D. [1986] : Matching Theory. Akadémiai Kiadó, Budapest 1986, and North-Holland, Amsterdam 1986

Schrijver, A. [1983] : Min-max results in combinatorial optimization ; Section 6. In : Mathematical Programming ; The State of the Art – Bonn 1982 (A. Bachem, M. Grötschel, B. Korte, eds.), Springer, Berlin 1983, pp. 439–500

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 29–33

Références citées :

Anstee, R.P. [1987] : A polynomial algorithm for b-matchings : an alternative approach. Information Processing Letters 24 (1987), 153–157

Caprara, A., Fischetti, M. [1996] : $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts. Mathematical Programming 74 (1996), 221–235

Edmonds, J. [1965] : Maximum matching and a polyhedron with (0,1) vertices. Journal of Research of the National Bureau of Standards B 69 (1965), 125–130

Edmonds, J., Johnson, E.L. [1970] : Matching : A well-solved class of integer linear programs. In : Combinatorial Structures and Their Applications ; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schonheim, eds.), Gordon and Breach, New York 1970, pp. 69–87

Edmonds, J., Johnson, E.L. [1973] : Matching, Euler tours and the Chinese postman problem. Mathematical Programming 5 (1973), 88–124

Guan, M. [1962] : Graphic programming using odd and even points. Chinese Mathematics 1 (1962), 273–277

Hadlock, F. [1975] : Finding a maximum cut of a planar graph in polynomial time. SIAM Journal on Computing 4 (1975), 221–225

Letchford, A.N., Reinelt, G., Theis, D.O. [2004] : A faster exact separation algorithm for blossom inequalities. Proceedings of the 10th Conference on Integer Programming and Combinatorial Optimization ; LNCS 3064 (D. Bienstock, G. Nemhauser, eds.), Springer, Berlin 2004, pp. 196–205

Marsh, A.B. [1979] : Matching algorithms. Ph.D. thesis, Johns Hopkins University, Baltimore 1979

Padberg, M.W., Rao, M.R. [1982] : Odd minimum cut-sets and b-matchings. Mathematics of Operations Research 7 (1982), 67–80

Pulleyblank, W.R. [1973] : Faces of matching polyhedra. Ph.D. thesis, University of Waterloo, 1973

Pulleyblank, W.R. [1980] : Dual integrality in b-matching problems. Mathematical Programming Study 12 (1980), 176–196

Rizzi, R. [2002] : Minimum T-cuts and optimal T-pairings. Discrete Mathematics 257 (2002), 177–181

Sebő, A. [1987] : A quick proof of Seymour's theorem on T-joins. Discrete Mathematics 64 (1987), 101–103

Seymour, P.D. [1981] : On odd cuts and multicommodity flows. Proceedings of the London Mathematical Society (3) 42 (1981), 178–192

Tutte, W.T. [1952] : The factors of graphs. Canadian Journal of Mathematics 4 (1952), 314–328

Tutte, W.T. [1954] : A short proof of the factor theorem for finite graphs. Canadian Journal of Mathematics 6 (1954), 347–352

Chapitre 13

Matroïdes

De nombreux problèmes d'optimisation combinatoire peuvent être formulés de la manière suivante. Soit un système d'ensembles $(E, \mathcal{F})$, c.-à-d. un ensemble fini E et un ensemble $\mathcal{F} \subseteq 2^E$, et une fonction coût $c : \mathcal{F} \to \mathbb{R}$; trouver un élément de $\mathcal{F}$ dont le coût soit minimum ou maximum. Par la suite, nous considérerons des fonctions modulaires c, c.-à-d. nous supposerons que $c(X) = c(\emptyset) + \sum_{x \in X}(c(\{x\}) - c(\emptyset))$ pour tout $X \subseteq E$. De manière équivalente, nous nous donnerons une fonction $c : E \to \mathbb{R}$ et écrirons $c(X) = \sum_{e \in X} c(e)$.

Dans ce chapitre, nous nous limitons aux problèmes d'optimisation combinatoire où $\mathcal{F}$ décrit un système d'indépendance (c.-à-d. $\mathcal{F}$ est fermé pour l'opération consistant à prendre des sous-ensembles) ou un matroïde. Les résultats de ce chapitre généralisent plusieurs résultats obtenus dans les chapitres précédents.

Au paragraphe 13.1, nous introduisons les systèmes d'indépendance et les matroïdes et nous montrons que de nombreux problèmes d'optimisation combinatoire peuvent être décrits dans ce contexte. Plusieurs systèmes d'axiomes équivalents permettent de définir les matroïdes (paragraphe 13.2) et une relation de dualité importante est présentée au paragraphe 13.3. La raison principale, justifiant l'importance des matroïdes, est qu'un simple algorithme glouton peut être utilisé pour l'optimisation sur les matroïdes. Nous analysons les algorithmes gloutons au paragraphe 13.4 avant de passer au problème de l'optimisation dans l'intersection de deux matroïdes. Nous prouvons aux paragraphes 13.5 et 13.7 que ce problème peut être résolu en temps polynomial. Nous montrons au paragraphe 13.6 que cela résout également le problème du recouvrement d'un matroïde par des ensembles indépendants.

13.1 Systèmes d'indépendance et matroïdes

Définition 13.1. *Un système d'ensembles $(E, \mathcal{F})$ est un* **système d'indépendance** *si :*

(M1) $\emptyset \in \mathcal{F}$;

(M2) *Si $X \subseteq Y \in \mathcal{F}$ alors $X \in \mathcal{F}$.*

Les éléments de $\mathcal{F}$ sont appelés **indépendants***, les éléments de $2^E \setminus \mathcal{F}$* **dépendants***. Les ensembles dépendants minimaux sont appelés* **circuits***, les ensembles indépendants maximaux sont appelés* **bases***. Pour $X \subseteq E$, les sous-ensembles indépendants maximaux de X sont appelés les bases de X.*

Définition 13.2. *Soit $(E, \mathcal{F})$ un système d'indépendance. Pour $X \subseteq E$, on définit le* **rang** *de X par $r(X) := \max\{|Y| : Y \subseteq X, Y \in \mathcal{F}\}$. De plus, on définit la* **fermeture** *de X par $\sigma(X) := \{y \in E : r(X \cup \{y\}) = r(X)\}$.*

Tout au long de ce chapitre, $(E, \mathcal{F})$ sera un système d'indépendance et $c : E \to \mathbb{R}$ sera une fonction coût. Nous nous concentrerons sur les deux types de problèmes suivants :

PROBLÈME DE MAXIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE	
Instance	Un système d'indépendance $(E, \mathcal{F})$ et $c : E \to \mathbb{R}$.
Tâche	Trouver un $X \in \mathcal{F}$ tel que $c(X) := \sum_{e \in X} c(e)$ soit maximum.

PROBLÈME DE MINIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE	
Instance	Un système d'indépendance $(E, \mathcal{F})$ et $c : E \to \mathbb{R}$.
Tâche	Trouver une base B telle que $c(B)$ soit minimum.

L'énoncé de ces problèmes est très général. L'ensemble E et la fonction coût c sont la plupart du temps décrits explicitement. En revanche, l'ensemble $\mathcal{F}$ n'est généralement pas décrit par une liste explicite de ses éléments. On suppose plutôt qu'un oracle permet de décider, pour un sous-ensemble donné $F \subseteq E$, si $F \in \mathcal{F}$. Nous reviendrons à cette question au paragraphe 13.4.

La liste suivante montre que de nombreux problèmes d'optimisation combinatoire peuvent être ramenés concrètement à l'une des deux formes précédentes :

(1) PROBLÈME DE L'ENSEMBLE STABLE DE POIDS MAXIMUM
Soient un graphe G et des poids $c : V(G) \to \mathbb{R}$, trouver un ensemble stable X dans G de poids maximum.
Ici $E = V(G)$ et $\mathcal{F} = \{F \subseteq E : F$ est stable dans $G\}$.

(2) PROBLÈME DU VOYAGEUR DE COMMERCE (PVC)
Soient un graphe complet non orienté G et des poids $c : E(G) \to \mathbb{R}_+$, trouver un cycle hamiltonien de poids minimum dans G.
Ici $E = E(G)$ et $\mathcal{F} = \{F \subseteq E : F$ est un sous-ensemble des arêtes d'un cycle hamiltonien de $G\}$.

(3) PROBLÈME DU PLUS COURT CHEMIN
Soient un graphe orienté G, $c : E(G) \to \mathbb{R}$ et $s, t \in V(G)$ tels que t soit connecté à s, trouver un plus court chemin de s à t dans G relativement à c.

Ici $E = E(G)$ et $\mathcal{F} = \{F \subseteq E : F$ est un sous-ensemble des arcs d'un chemin de s à $t\}$.

(4) PROBLÈME DU SAC À DOS
Soient des nombres positifs ou nuls n, c_i, w_i $(1 \leq i \leq n)$, et W, trouver un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} w_j \leq W$ et tel que $\sum_{j \in S} c_j$ soit maximum.
Ici $E = \{1, \ldots, n\}$ et $\mathcal{F} = \left\{F \subseteq E : \sum_{j \in F} w_j \leq W\right\}$.

(5) PROBLÈME DE L'ARBRE COUVRANT MINIMUM
Soient un graphe connexe non orienté G et des poids $c : E(G) \to \mathbb{R}$, trouver un arbre couvrant de poids minimum dans G.
Ici $E = E(G)$ et $\mathcal{F}$ est l'ensemble des forêts de G.

(6) PROBLÈME DE LA FORÊT DE POIDS MAXIMUM
Soient un graphe non orienté G et des poids $c : E(G) \to \mathbb{R}$, trouver une forêt de poids maximum dans G.
Ici encore $E = E(G)$ et $\mathcal{F}$ est l'ensemble des forêts de G.

(7) PROBLÈME DE L'ARBRE DE STEINER
Soient un graphe connexe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$, et un ensemble $T \subseteq V(G)$ de nœuds terminaux, trouver un arbre de Steiner pour T, c.-à-d. un arbre S avec $T \subseteq V(S)$ et $E(S) \subseteq E(G)$, tel que $c(E(S))$ soit minimum.
Ici $E = E(G)$ et $\mathcal{F} = \{F \subseteq E : F$ est un sous-ensemble d'un arbre de Steiner pour $T\}$.

(8) PROBLÈME DE LA RAMIFICATION DE POIDS MAXIMUM
Soient un graphe orienté G et des poids $c : E(G) \to \mathbb{R}$, trouver une ramification de poids maximum dans G.
Ici $E = E(G)$ et $\mathcal{F}$ est l'ensemble des ramifications de G.

(9) PROBLÈME DU COUPLAGE DE POIDS MAXIMUM
Soient un graphe non orienté G et des poids $c : E(G) \to \mathbb{R}$, trouver un couplage de poids maximum dans G.
Ici $E = E(G)$ et $\mathcal{F}$ est l'ensemble des couplages de G.

Cette liste contient des problèmes *NP*-difficiles ((1),(2),(4),(7)) ainsi que des problèmes polynomiaux ((5),(6),(8),(9)). Sous la forme précédente, le problème (3) est *NP*-difficile, mais il est polynomial si les poids sont de plus supposés positifs ou nuls. (La notion de problème *NP*-difficile est introduite au chapitre 15.)

Définition 13.3. *Un système d'indépendance est un* **matroïde** *si*

(M3) *Si $X, Y \in \mathcal{F}$ et $|X| > |Y|$, alors il existe un $x \in X \setminus Y$ tel que $Y \cup \{x\} \in \mathcal{F}$.*

Le terme de matroïde souligne le fait que cette structure est une généralisation des matrices. Cela apparaîtra clairement dans le premier exemple de la proposition suivante :

Proposition 13.4. *Les systèmes d'indépendance $(E, \mathcal{F})$ suivants sont des matroïdes :*

(a) *E est un ensemble de colonnes d'une matrice A définie sur un corps K, et $\mathcal{F} := \{F \subseteq E$: les colonnes de F sont linéairement indépendantes sur $K\}$.*

(b) *E est un ensemble d'arêtes d'un graphe non orienté G et $\mathcal{F} := \{F \subseteq E : (V(G), F)$ est une forêt$\}$.*

(c) *E est un ensemble fini, k est un entier et $\mathcal{F} := \{F \subseteq E : |F| \leq k\}$.*

(d) *E est un ensemble d'arêtes d'un graphe non orienté G, S est un ensemble stable de G, k_s sont des entiers ($s \in S$) et $\mathcal{F} := \{F \subseteq E : |\delta_F(s)| \leq k_s$ pour tout $s \in S\}$.*

(e) *E est un ensemble d'arcs d'un graphe orienté G, $S \subseteq V(G)$, k_s sont des entiers ($s \in S$) et $\mathcal{F} := \{F \subseteq E : |\delta_F^-(s)| \leq k_s$ pour tout $s \in S\}$.*

Preuve. Dans chaque cas, il est évident que $(E, \mathcal{F})$ est bien un système d'indépendance. Il reste donc à prouver que (M3) est bien réalisé. Pour (a), c'est un résultat connu d'algébre linéaire. Pour (c), cela est évident.

Afin de prouver (M3) pour (b), considérons $X, Y \in \mathcal{F}$ et supposons $Y \cup \{x\} \notin \mathcal{F}$ pour tout $x \in X \setminus Y$. Montrons qu'alors $|X| \leq |Y|$. Pour chaque arête $x = (v, w) \in X$, v et w sont dans la même composante connexe de $(V(G), Y)$. Ainsi, chaque composante connexe $Z \subseteq V(G)$ de $(V(G), X)$ est un sous-ensemble d'une composante connexe de $(V(G), Y)$. Le nombre p de composantes connexes de la forêt $(V(G), X)$ est donc supérieur ou égal au nombre q de composantes connexes de la forêt $(V(G), Y)$. Mais alors $|V(G)| - |X| = p \geq q = |V(G)| - |Y|$, ce qui implique $|X| \leq |Y|$.

Pour vérifier (M3) pour (d), considérons $X, Y \in \mathcal{F}$ tels que $|X| > |Y|$. Soit $S' := \{s \in S : |\delta_Y(s)| = k_s\}$. Comme $|X| > |Y|$ et $|\delta_X(s)| \leq k_s$ pour tout $s \in S'$, il existe un $e \in X \setminus Y$ tel que $e \notin \delta(s)$ pour $s \in S'$. Alors $Y \cup \{e\} \in \mathcal{F}$.

Pour (e), la preuve est identique en remplaçant δ par δ^-. □

On donne des noms spécifiques à certains de ces matroïdes : le matroïde défini en (a) est appelé le **matroïde vectoriel** associé à A. Soit $\mathcal{M}$ un matroïde. S'il existe une matrice A sur le corps F telle que $\mathcal{M}$ est le matroïde vectoriel associé à A, alors $\mathcal{M}$ est dit **représentable sur** F. Il existe des matroïdes qui ne sont pas représentables.

Le matroïde défini en (b) est appelé le **matroïde des cycles de** G et sera parfois noté $\mathcal{M}(G)$. Un matroïde est appelé **matroïde graphique** si c'est le matroïde des cycles d'un graphe.

Le matroïde défini en (c) est appelé un **matroïde uniforme**.

Parmi les systèmes d'indépendance donnés dans la liste au début de ce paragraphe, seuls les systèmes (5) et (6) correspondent à des matroïdes et plus précisément à des matroïdes graphiques. Le théorème suivant permet de prouver facilement que les autres systèmes d'indépendance de la liste précédente ne sont généralement pas des matroïdes (exercice 1) :

Théorème 13.5. *Soit $(E, \mathcal{F})$ un système d'indépendance. Alors les affirmations suivantes sont équivalentes :*

(M3) *Si $X, Y \in \mathcal{F}$ et $|X| > |Y|$, alors il existe un $x \in X \setminus Y$ tel que $Y \cup \{x\} \in \mathcal{F}$.*

(M3′) *Si $X, Y \in \mathcal{F}$ et $|X| = |Y| + 1$, alors il existe un $x \in X \setminus Y$ tel que $Y \cup \{x\} \in \mathcal{F}$.*

(M3″) *Pour tout $X \subseteq E$, toutes les bases de X ont la même cardinalité.*

Preuve. Trivialement, (M3)⇔(M3′) et (M3)⇒(M3″). Afin de prouver (M3″)⇒(M3), considérons $X, Y \in \mathcal{F}$ tels que $|X| > |Y|$. D'après (M3″), Y ne peut pas être une base de $X \cup Y$. Il doit donc exister un $x \in (X \cup Y) \setminus Y = X \setminus Y$ tel que $Y \cup \{x\} \in \mathcal{F}$. □

Il est parfois utile de considérer une seconde fonction rang :

Définition 13.6. *Soit $(E, \mathcal{F})$ un système d'indépendance. Pour $X \subseteq E$, on définit le rang inférieur par*

$$\rho(X) := \min\{|Y| : Y \subseteq X, Y \in \mathcal{F} \text{ et } Y \cup \{x\} \notin \mathcal{F} \text{ pour tout } x \in X \setminus Y\}.$$

Le **rang quotient** *de $(E, \mathcal{F})$ est défini par*

$$q(E, \mathcal{F}) \;:=\; \min_{F \subseteq E} \frac{\rho(F)}{r(F)} \,.$$

Proposition 13.7. *Soit $(E, \mathcal{F})$ un système d'indépendance. Alors $q(E, \mathcal{F}) \leq 1$. De plus, $(E, \mathcal{F})$ est un matroïde si et seulement si $q(E, \mathcal{F}) = 1$.*

Preuve. $q(E, \mathcal{F}) \leq 1$ par définition du rang quotient. $q(E, \mathcal{F}) = 1$ est évidemment équivalent à (M3″). □

Afin d'évaluer le rang quotient, on peut utiliser le résultat suivant :

Théorème 13.8. (Hausmann, Jenkyns et Korte [1980]) *Soit $(E, \mathcal{F})$ un système d'indépendance. Si, pour tout $A \in \mathcal{F}$ et pour tout $e \in E$, $A \cup \{e\}$ contient au plus p circuits, alors $q(E, \mathcal{F}) \geq \frac{1}{p}$.*

Preuve. Soit $F \subseteq E$ et J, K deux bases de F. Nous prouvons l'inégalité $\frac{|J|}{|K|} \geq \frac{1}{p}$.

Soit $J \setminus K = \{e_1, \ldots, e_t\}$. Nous construisons une suite $K = K_0, K_1, \ldots, K_t$ de sous-ensembles indépendants de $J \cup K$ tels que $J \cap K \subseteq K_i$, $K_i \cap \{e_1, \ldots, e_t\} = \{e_1, \ldots, e_i\}$ et $|K_{i-1} \setminus K_i| \leq p$ pour $i = 1, \ldots, t$.

Puisque $K_i \cup \{e_{i+1}\}$ contient au plus p circuits et que chacun de ces circuits doit rencontrer $K_i \setminus J$ (car J est indépendant), il existe un sous-ensemble $X \subseteq K_i \setminus J$ tel que $|X| \leq p$ et $(K_i \setminus X) \cup \{e_{i+1}\} \in \mathcal{F}$. Nous posons $K_{i+1} := (K_i \setminus X) \cup \{e_{i+1}\}$.

Alors $J \subseteq K_t \in \mathcal{F}$. Comme J est une base de F, $J = K_t$. Nous avons donc

$$|K \setminus J| \;=\; \sum_{i=1}^{t} |K_{i-1} \setminus K_i| \;\leq\; pt \;=\; p\,|J \setminus K|,$$

ce qui prouve $|K| \leq p\,|J|$. □

Ce résultat montre que, pour l'exemple (9), nous avons $q(E,\mathcal{F}) \geq \frac{1}{2}$ (voir également exercice 1 du chapitre 10). En fait $q(E,\mathcal{F}) = \frac{1}{2}$ si et seulement si G contient une chaîne de longueur 3 comme sous-graphe (sinon $q(E,\mathcal{F}) = 1$). Pour l'exemple (1) de notre liste, le rang quotient du système d'indépendance peut être arbitrairement petit (considérer le cas où G est un graphe étoile). Nous renvoyons l'étude des rangs quotients des autres systèmes d'indépendance à l'exercice 5.

13.2 Autres axiomes

Dans ce paragraphe, nous présentons d'autres systèmes d'axiomes définissant les matroïdes. Ils caractérisent les propriétés fondamentales de l'ensemble des bases, de la fonction rang, de l'opération de fermeture et de l'ensemble des circuits d'un matroïde.

Théorème 13.9. *Soit E un ensemble fini et $\mathcal{B} \subseteq 2^E$. $\mathcal{B}$ est l'ensemble des bases d'un matroïde $(E,\mathcal{F})$ si et seulement si :*

(B1) $\mathcal{B} \neq \emptyset$ *;*

(B2) *Pour tout $B_1, B_2 \in \mathcal{B}$ et tout $x \in B_1 \setminus B_2$ il existe un $y \in B_2 \setminus B_1$ tel que $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.*

Preuve. L'ensemble des bases d'un matroïde vérifie (B1) (d'après (M1)) et (B2) : soient B_1, B_2 deux bases et soit $x \in B_1 \setminus B_2$; $B_1 \setminus \{x\}$ est indépendant et d'après (M3) il existe un $y \in B_2 \setminus B_1$ tel que $(B_1 \setminus \{x\}) \cup \{y\}$ est indépendant. Alors $(B_1 \setminus \{x\}) \cup \{y\}$ doit bien être une base puisque toutes les bases d'un matroïde ont la même cardinalité,

Pour le sens inverse, soit $\mathcal{B}$ vérifiant (B1) et (B2), nous montrons d'abord que tous les éléments de $\mathcal{B}$ ont la même cardinalité : sinon considérons $B_1, B_2 \in \mathcal{B}$ tels que $|B_1| > |B_2|$ et tels que $|B_1 \cap B_2|$ soit maximum. Soit $x \in B_1 \setminus B_2$. D'après (B2) il existe un $y \in B_2 \setminus B_1$ tel que $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$, ce qui contredit la maximalité de $|B_1 \cap B_2|$.

Considérons alors

$$\mathcal{F} := \{F \subseteq E : \text{il existe un } B \in \mathcal{B} \text{ tel que } F \subseteq B\}.$$

$(E,\mathcal{F})$ est un système d'indépendance, et $\mathcal{B}$ est l'ensemble de ses bases. Afin de montrer que $(E,\mathcal{F})$ vérifie (M3), considérons $X, Y \in \mathcal{F}$ tels que $|X| > |Y|$. Soient $X \subseteq B_1 \in \mathcal{B}$ et $Y \subseteq B_2 \in \mathcal{B}$, où B_1 et B_2 sont choisis de sorte que $|B_1 \cap B_2|$ soit maximum. Si $B_2 \cap (X \setminus Y) \neq \emptyset$, la preuve est terminée, car on peut augmenter Y.

Montrons que le cas où $B_2 \cap (X \setminus Y) = \emptyset$ est impossible. En effet, sous cette hypothèse on obtient

$$|B_1 \cap B_2| + |Y \setminus B_1| + |(B_2 \setminus B_1) \setminus Y| \;=\; |B_2| \;=\; |B_1| \;\geq\; |B_1 \cap B_2| + |X \setminus Y|.$$

Comme $|X \setminus Y| > |Y \setminus X| \geq |Y \setminus B_1|$, cela implique $(B_2 \setminus B_1) \setminus Y \neq \emptyset$. Considérons alors $y \in (B_2 \setminus B_1) \setminus Y$. D'après (B2) il existe un $x \in B_1 \setminus B_2$ tel que $(B_2 \setminus \{y\}) \cup \{x\} \in \mathcal{B}$, ce qui contredit la maximalité de $|B_1 \cap B_2|$. □

Voir l'exercice 7 pour un résultat similaire. Une propriété très importante des matroïdes est que la fonction rang est sous-modulaire :

Théorème 13.10. *Soient E un ensemble fini et $r : 2^E \to \mathbb{Z}_+$. Alors les affirmations suivantes sont équivalentes :*

(a) *r est la fonction rang d'un matroïde $(E, \mathcal{F})$ (où $\mathcal{F} = \{F \subseteq E : r(F) = |F|\}$).*

(b) *Pour tout $X, Y \subseteq E$:*

(R1) $r(X) \leq |X|$;

(R2) *Si $X \subseteq Y$ alors* $r(X) \leq r(Y)$;

(R3) $r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y)$.

(c) *Pour tout $X \subseteq E$ et $x, y \in E$:*

(R1′) $r(\emptyset) = 0$;

(R2′) $r(X) \leq r(X \cup \{y\}) \leq r(X) + 1$;

(R3′) *Si* $r(X \cup \{x\}) = r(X \cup \{y\}) = r(X)$ *alors* $r(X \cup \{x, y\}) = r(X)$.

Preuve. (a)⇒(b) : si r est la fonction rang d'un système d'indépendance $(E, \mathcal{F})$, (R1) et (R2) sont évidemment vérifiées. Si $(E, \mathcal{F})$ est un matroïde, on peut aussi prouver (R3) :

Soient $X, Y \subseteq E$, et soit A une base de $X \cap Y$. D'après (M3), A peut être prolongée en une base $A \dot\cup B$ de X et en une base $(A \cup B) \dot\cup C$ de $X \cup Y$. Alors $A \cup C$ est un sous-ensemble indépendant de Y, donc

$$\begin{aligned} r(X) + r(Y) &\geq |A \cup B| + |A \cup C| \\ &= 2|A| + |B| + |C| \\ &= |A \cup B \cup C| + |A| \\ &= r(X \cup Y) + r(X \cap Y). \end{aligned}$$

(b)⇒(c) : (R1) implique (R1′). $r(X) \leq r(X \cup \{y\})$ se déduit de (R2). D'après (R3) et (R1),

$$r(X \cup \{y\}) \;\leq\; r(X) + r(\{y\}) - r(X \cap \{y\}) \;\leq\; r(X) + r(\{y\}) \;\leq\; r(X) + 1,$$

ce qui prouve (R2′).

(R3′) est trivial pour $x = y$. Pour $x \neq y$ on a, d'après (R2) et (R3),

$$2r(X) \;\leq\; r(X) + r(X \cup \{x, y\}) \;\leq\; r(X \cup \{x\}) + r(X \cup \{y\}),$$

ce qui implique (R3′).

(c)⇒(a) : soit $r : 2^E \to \mathbb{Z}_+$ une fonction vérifiant (R1′)–(R3′). Soit

$$\mathcal{F} := \{F \subseteq E : r(F) = |F|\}.$$

Nous affirmons que $(E, \mathcal{F})$ est un matroïde. (M1) se déduit de (R1′). (R2′) implique $r(X) \leq |X|$ pour tout $X \subseteq E$. Si $Y \in \mathcal{F}$, $y \in Y$ et $X := Y \setminus \{y\}$, nous avons

$$|X| + 1 \;=\; |Y| \;=\; r(Y) \;=\; r(X \cup \{y\}) \;\leq\; r(X) + 1 \;\leq\; |X| + 1,$$

donc $X \in \mathcal{F}$. Cela implique (M2).

Soient alors $X, Y \in \mathcal{F}$ tels que $|X| = |Y| + 1$. Soit $X \setminus Y = \{x_1, \ldots, x_k\}$. Supposons que (M3′) soit violée, c.-à-d. $r(Y \cup \{x_i\}) = |Y|$ pour $i = 1, \ldots, k$. Alors d'après (R3′) $r(Y \cup \{x_1, x_i\}) = r(Y)$ pour $i = 2, \ldots, k$. L'application répétée de cet argument donne $r(Y) = r(Y \cup \{x_1, \ldots, x_k\}) = r(X \cup Y) \geq r(X)$, ce qui fournit une contradiction.

Donc $(E, \mathcal{F})$ est bien un matroïde. Afin de montrer que r est la fonction rang de ce matroïde, nous devons prouver que $r(X) = \max\{|Y| : Y \subseteq X,\, r(Y) = |Y|\}$ pour tout $X \subseteq E$. Soit donc $X \subseteq E$, et soit Y un sous-ensemble maximum de X tel que $r(Y) = |Y|$. Pour tout $x \in X \setminus Y$ nous avons $r(Y \cup \{x\}) < |Y| + 1$, donc d'après (R2′) $r(Y \cup \{x\}) = |Y|$. L'application répétée de (R3′) implique $r(X) = |Y|$. □

Théorème 13.11. *Soit E un ensemble fini et soit une fonction $\sigma : 2^E \to 2^E$. σ est l'opérateur de fermeture d'un matroïde $(E, \mathcal{F})$ si et seulement si les conditions suivantes sont vérifiées pour tout $X, Y \subseteq E$ et pour tout $x, y \in E$:*

(S1) $X \subseteq \sigma(X)$ *;*

(S2) $X \subseteq Y \subseteq E$ *implique* $\sigma(X) \subseteq \sigma(Y)$ *;*

(S3) $\sigma(X) = \sigma(\sigma(X))$ *;*

(S4) *Si* $y \notin \sigma(X)$ *et* $y \in \sigma(X \cup \{x\})$ *alors* $x \in \sigma(X \cup \{y\})$.

Preuve. Si σ est l'opérateur de fermeture d'un matroïde, alors (S1) est trivialement vérifié.

Pour $X \subseteq Y$ et $z \in \sigma(X)$, nous avons, d'après (R3) et (R2),

$$\begin{aligned} r(X) + r(Y) &= r(X \cup \{z\}) + r(Y) \\ &\geq r((X \cup \{z\}) \cap Y) + r(X \cup \{z\} \cup Y) \\ &\geq r(X) + r(Y \cup \{z\}), \end{aligned}$$

ce qui implique $z \in \sigma(Y)$ et prouve donc (S2).

Par l'application répétée de (R3′), nous avons $r(\sigma(X)) = r(X)$ pour tout X, ce qui implique (S3).

Afin de prouver (S4), supposons qu'il existe X, x, y tels que $y \notin \sigma(X)$, $y \in \sigma(X \cup \{x\})$ et $x \notin \sigma(X \cup \{y\})$. Alors $r(X \cup \{y\}) = r(X) + 1$, $r(X \cup \{x, y\}) = r(X \cup \{x\})$ et $r(X \cup \{x, y\}) = r(X \cup \{y\}) + 1$. Ainsi $r(X \cup \{x\}) = r(X) + 2$, ce qui contredit (R2′).

Pour montrer le sens inverse, considérons une fonction $\sigma : 2^E \to 2^E$ vérifiant (S1)–(S4). Soit

$$\mathcal{F} := \{X \subseteq E : x \notin \sigma(X \setminus \{x\}) \text{ pour tout } x \in X\}.$$

Nous affirmons que $(E, \mathcal{F})$ est un matroïde.

(M1) est évident. Pour $X \subseteq Y \in \mathcal{F}$ et $x \in X$, nous avons $x \notin \sigma(Y \setminus \{x\}) \supseteq \sigma(X \setminus \{x\})$, donc $X \in \mathcal{F}$ et (M2) est vérifié. Afin de prouver (M3) nous avons besoin du résultat suivant :

Affirmation : pour $X \in \mathcal{F}$ et $Y \subseteq E$ tels que $|X| > |Y|$ nous avons $X \not\subseteq \sigma(Y)$.

Nous prouvons cette affirmation par induction sur $|Y \setminus X|$. Si $Y \subset X$, alors considérons $x \in X \setminus Y$. Puisque $X \in \mathcal{F}$ nous avons $x \notin \sigma(X \setminus \{x\}) \supseteq \sigma(Y)$ d'après (S2). Par conséquent $x \in X \setminus \sigma(Y)$ comme souhaité.

Si $|Y \setminus X| > 0$, alors considérons $y \in Y \setminus X$. Par l'hypothèse d'induction, il existe un $x \in X \setminus \sigma(Y \setminus \{y\})$. Si $x \notin \sigma(Y)$, l'affirmation est démontrée. Sinon $x \notin \sigma(Y \setminus \{y\})$, mais $x \in \sigma(Y) = \sigma((Y \setminus \{y\}) \cup \{y\})$, donc d'après (S4) $y \in \sigma((Y \setminus \{y\}) \cup \{x\})$. D'après (S1), nous obtenons $Y \subseteq \sigma((Y \setminus \{y\}) \cup \{x\})$ et ainsi $\sigma(Y) \subseteq \sigma((Y \setminus \{y\}) \cup \{x\})$ d'après (S2) et (S3). En appliquant l'hypothèse d'induction à X et $(Y \setminus \{y\}) \cup \{x\}$ (remarquer que $x \neq y$), nous obtenons $X \not\subseteq \sigma((Y \setminus \{y\}) \cup \{x\})$, donc $X \not\subseteq \sigma(Y)$ comme souhaité.

L'affirmation étant démontrée, nous pouvons facilement vérifier (M3). Soient $X, Y \in \mathcal{F}$ tels que $|X| > |Y|$. D'après l'affirmation il existe un $x \in X \setminus \sigma(Y)$. Pour tout $z \in Y \cup \{x\}$, nous avons alors $z \notin \sigma(Y \setminus \{z\})$, car $Y \in \mathcal{F}$ et $x \notin \sigma(Y) = \sigma(Y \setminus \{x\})$. D'après (S4), $z \notin \sigma(Y \setminus \{z\})$ et $x \notin \sigma(Y)$ impliquent $z \notin \sigma((Y \setminus \{z\}) \cup \{x\}) \supseteq \sigma((Y \cup \{x\}) \setminus \{z\})$. Ainsi $Y \cup \{x\} \in \mathcal{F}$.

Donc (M3) est bien vérifié et $(E, \mathcal{F})$ est un matroïde. Notons r la fonction rang et σ' l'opérateur de fermeture associés à $(E, \mathcal{F})$. Il reste à prouver que $\sigma = \sigma'$.

Par définition, $\sigma'(X) = \{y \in E : r(X \cup \{y\}) = r(X)\}$ et

$$r(X) = \max\{|Y| : Y \subseteq X,\ y \notin \sigma(Y \setminus \{y\}) \text{ pour tout } y \in Y\}$$

pour tout $X \subseteq E$.

Soit $X \subseteq E$. Afin de montrer $\sigma'(X) \subseteq \sigma(X)$, considérons $z \in \sigma'(X) \setminus X$. Soit Y une base de X. Puisque $r(Y \cup \{z\}) \leq r(X \cup \{z\}) = r(X) = |Y| < |Y \cup \{z\}|$ il existe un $y \in Y \cup \{z\}$ tel que $y \in \sigma((Y \cup \{z\}) \setminus \{y\})$. Si $y = z$, alors nous avons $z \in \sigma(Y)$. Sinon, comme $y \notin \sigma(Y \setminus \{y\})$, nous avons d'après (S4), $z \in \sigma(Y)$. Alors, d'après (S2), $z \in \sigma(X)$. De plus, d'après (S1), si $z \in X$ alors $z \in \sigma(X)$. Cela implique $\sigma'(X) \subseteq \sigma(X)$.

Considérons alors $z \notin \sigma'(X)$, c.-à-d. $r(X \cup \{z\}) > r(X)$. Soit Y une base de $X \cup \{z\}$. Alors $z \in Y$ et $|Y \setminus \{z\}| = |Y| - 1 = r(X \cup \{z\}) - 1 = r(X)$. Par conséquent, $Y \setminus \{z\}$ est une base de X, ce qui implique $X \subseteq \sigma'(Y \setminus \{z\}) \subseteq \sigma(Y \setminus \{z\})$ et donc $\sigma(X) \subseteq \sigma(Y \setminus \{z\})$. Comme $z \notin \sigma(Y \setminus \{z\})$, nous en concluons que $z \notin \sigma(X)$. □

Théorème 13.12. *Soit E un ensemble fini et soit $\mathcal{C} \subseteq 2^E$. Nous dirons que $\mathcal{C}$ est l'ensemble des circuits d'un système d'indépendance $(E, \mathcal{F})$, avec $\mathcal{F} = \{F \subset E :$*

il n'existe aucun $C \in \mathcal{C}$ tel que $C \subseteq F$}, si et seulement si les conditions suivantes sont vérifiées :

(C1) $\emptyset \notin \mathcal{C}$;

(C2) *pour tout $C_1, C_2 \in \mathcal{C}$, $C_1 \subseteq C_2$ implique $C_1 = C_2$.*

De plus, si $\mathcal{C}$ est l'ensemble des circuits d'un système d'indépendance $(E, \mathcal{F})$, alors les affirmations suivantes sont équivalentes :

(a) *$(E, \mathcal{F})$ est un matroïde.*

(b) *pour tout $X \in \mathcal{F}$ et $e \in E$, $X \cup \{e\}$ contient au plus un circuit.*

(C3) *Pour tout $C_1, C_2 \in \mathcal{C}$ tels que $C_1 \neq C_2$ et $e \in C_1 \cap C_2$ il existe $C_3 \in \mathcal{C}$ tel que $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.*

(C3′) *Pour tout $C_1, C_2 \in \mathcal{C}$, $e \in C_1 \cap C_2$ et $f \in C_1 \setminus C_2$, il existe $C_3 \in \mathcal{C}$ tel que $f \in C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.*

Preuve. Par définition, la famille des circuits d'un système d'indépendance vérifie (C1) et (C2). Si $\mathcal{C}$ vérifie (C1), alors $(E, \mathcal{F})$ est un système d'indépendance. Si $\mathcal{C}$ vérifie aussi (C2), c'est l'ensemble des circuits de ce système d'indépendance.

(a)⇒(C3′) : soit $\mathcal{C}$ la famille des circuits d'un matroïde, et soit $C_1, C_2 \in \mathcal{C}$, $e \in C_1 \cap C_2$ et $f \in C_1 \setminus C_2$. En appliquant deux fois (R3), nous avons

$$\begin{aligned} & |C_1| - 1 + r((C_1 \cup C_2) \setminus \{e, f\}) + |C_2| - 1 \\ = \; & r(C_1) + r((C_1 \cup C_2) \setminus \{e, f\}) + r(C_2) \\ \geq \; & r(C_1) + r((C_1 \cup C_2) \setminus \{f\}) + r(C_2 \setminus \{e\}) \\ \geq \; & r(C_1 \setminus \{f\}) + r(C_1 \cup C_2) + r(C_2 \setminus \{e\}) \\ = \; & |C_1| - 1 + r(C_1 \cup C_2) + |C_2| - 1. \end{aligned}$$

Donc $r((C_1 \cup C_2) \setminus \{e, f\}) = r(C_1 \cup C_2)$. Soit B une base de $(C_1 \cup C_2) \setminus \{e, f\}$. Alors $B \cup \{f\}$ contient un circuit C_3, tel que $f \in C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$ comme souhaité.

(C3′)⇒(C3) : évident.

(C3)⇒(b) : si $X \in \mathcal{F}$ et $X \cup \{e\}$ contient deux circuits C_1, C_2, (C3) implique $(C_1 \cup C_2) \setminus \{e\} \notin \mathcal{F}$. Mais $(C_1 \cup C_2) \setminus \{e\}$ est un sous-ensemble de X.

(b)⇒(a) : se déduit du théorème 13.8 et de la proposition 13.7. □

En particulier la propriété (b) sera souvent utilisée. Pour $X \in \mathcal{F}$ et $e \in E$ tels que $X \cup \{e\} \notin \mathcal{F}$, nous noterons $C(X, e)$ l'unique circuit contenu dans $X \cup \{e\}$. Si $X \cup \{e\} \in \mathcal{F}$, nous noterons $C(X, e) := \emptyset$.

13.3 Dualité

La dualité est une des notions fondamentales de la théorie des matroïdes.

Définition 13.13. *Soit $(E, \mathcal{F})$ un système d'indépendance. Nous définissons le* **dual** *de $(E, \mathcal{F})$ par $(E, \mathcal{F}^*)$, où*

$$\mathcal{F}^* \;=\; \{F \subseteq E :\ \textit{il existe une base } B \textit{ de } (E, \mathcal{F}) \textit{ telle que } F \cap B = \emptyset\}.$$

Il est évident que le dual d'un système d'indépendance est aussi un système d'indépendance.

Proposition 13.14. $(E, \mathcal{F}^{**}) = (E, \mathcal{F})$.

Preuve. $F \in \mathcal{F}^{**} \Leftrightarrow$ il existe une base B^* de $(E, \mathcal{F}^*)$ telle que $F \cap B^* = \emptyset \Leftrightarrow$ il existe une base B de $(E, \mathcal{F})$ telle que $F \cap (E \setminus B) = \emptyset \Leftrightarrow F \in \mathcal{F}$. □

Théorème 13.15. *Soient $(E, \mathcal{F})$ un système d'indépendance, $(E, \mathcal{F}^*)$ son dual, et soient r et r^* leurs fonctions rangs respectives.*

(a) *$(E, \mathcal{F})$ est un matroïde si et seulement si $(E, \mathcal{F}^*)$ est un matroïde.* (Whitney [1935])

(b) *Si $(E, \mathcal{F})$ est un matroïde, alors $r^*(F) = |F| + r(E \setminus F) - r(E)$ pour $F \subseteq E$.*

Preuve. D'après la proposition 13.14, nous devons seulement prouver un des sens pour (a). Considérons donc un matroïde $(E, \mathcal{F})$. Nous définissons $q : 2^E \to \mathbb{Z}_+$ par $q(F) := |F| + r(E \setminus F) - r(E)$. Nous affirmons que q vérifie (R1), (R2) et (R3). Alors, d'après le théorème 13.10, q est la fonction rang d'un matroïde. Évidemment $q(F) = |F|$ si et seulement si $F \in \mathcal{F}^*$. Nous en concluons que $q = r^*$, et (a) et (b) sont prouvés.

Prouvons alors l'affirmation précédente : q vérifie (R1), car r vérifie (R2). Pour vérifier que q satisfait (R2), considérons $X \subseteq Y \subseteq E$. Puisque $(E, \mathcal{F})$ est un matroïde, r vérifie (R3), donc

$$r(E \setminus X) + 0 \;=\; r((E \setminus Y) \cup (Y \setminus X)) + r(\emptyset) \;\leq\; r(E \setminus Y) + r(Y \setminus X).$$

Nous en concluons que

$$r(E \setminus X) - r(E \setminus Y) \;\leq\; r(Y \setminus X) \;\leq\; |Y \setminus X| \;=\; |Y| - |X|$$

(remarquer que r vérifie (R1)), donc $q(X) \leq q(Y)$.

Il reste à prouver que q vérifie (R3). Considérons $X, Y \subseteq E$. Comme r vérifie (R3) nous avons

$$\begin{aligned}
& q(X \cup Y) + q(X \cap Y) \\
= \;& |X \cup Y| + |X \cap Y| + r(E \setminus (X \cup Y)) + r(E \setminus (X \cap Y)) - 2r(E) \\
= \;& |X| + |Y| + r((E \setminus X) \cap (E \setminus Y)) + r((E \setminus X) \cup (E \setminus Y)) - 2r(E) \\
\leq \;& |X| + |Y| + r(E \setminus X) + r(E \setminus Y) - 2r(E) \\
= \;& q(X) + q(Y).
\end{aligned}$$

□

Pour tout graphe G, nous avons défini le matroïde des cycles $\mathcal{M}(G)$ qui a bien sûr un dual. Pour un graphe planaire G, il existe aussi un graphe planaire dual G^* (qui en général dépend de la représentation de G). Il est intéressant de noter que ces deux notions de dualité coïncident :

Théorème 13.16. *Soient G un graphe planaire connexe avec une représentation plane arbitraire, et G^* le graphe planaire dual. Alors*

$$\mathcal{M}(G^*) \;=\; (\mathcal{M}(G))^*.$$

Preuve. Pour $T \subseteq E(G)$ nous notons $\overline{T}^* := \{e^* : e \in E(G) \setminus T\}$, où e^* est le dual de l'arête e. Nous devons prouver l'affirmation suivante :
Affirmation : T est l'ensemble des arêtes d'un arbre couvrant de G si et seulement si $\overline{T}^*$ est l'ensemble des arêtes d'un arbre couvrant de G^*.

Comme $(G^*)^* = G$ (d'après la proposition 2.42) et que $\overline{(\overline{T}^*)}^* = T$, il suffit de prouver l'un des sens de l'affirmation.

Considérons donc $T \subseteq E(G)$, tel que $\overline{T}^*$ soit l'ensemble des arêtes d'un arbre couvrant de G^*. $(V(G), T)$ doit être connexe, sinon une composante connexe définirait une coupe dont le dual contiendrait un circuit dans $\overline{T}^*$ (théorème 2.43). D'autre part, si $(V(G), T)$ contient un cycle, alors l'ensemble d'arêtes dual correspond à une coupe et $(V(G^*), \overline{T}^*)$ n'est pas connexe. Ainsi $(V(G), T)$ est bien un arbre couvrant de G. □

Cela implique que si G est planaire alors $(\mathcal{M}(G))^*$ est un matroïde graphique. Si, pour tout graphe G, $(\mathcal{M}(G))^*$ est un matroïde graphique, disons par exemple $(\mathcal{M}(G))^* = \mathcal{M}(G')$, alors G' est évidemment un dual abstrait de G. D'après l'exercice 34 du chapitre 2, l'inverse est également vrai : G est un graphe planaire si et seulement si G a un dual abstrait (Whitney [1933]). Cela implique que $(\mathcal{M}(G))^*$ est un matroïde graphique si et seulement si G est un graphe planaire.

Remarquons que le théorème 13.16 implique de manière assez immédiate la formule d'Euler (théorème 2.32) : soit G un graphe planaire connexe et une représentation plane de ce graphe, et soit $\mathcal{M}(G)$ le matroïde des cycles de G. D'après le théorème 13.15 (b), $r(E(G))+r^*(E(G)) = |E(G)|$. Comme $r(E(G)) = |V(G)| - 1$ (le nombre d'arêtes d'un arbre couvrant) et comme $r^*(E(G)) = |V(G^*)| - 1$ (d'après le théorème 13.16), nous obtenons que le nombre de faces de G est $|V(G^*)| = |E(G)| - |V(G)| + 2$, c.-à-d. la formule d'Euler.

La dualité des systèmes d'indépendance a également d'intéressantes applications en combinatoire polyédrale. Un système d'ensemble $(E, \mathcal{F})$ est un **amas** (en anglais *clutter*) si $X \not\subset Y$ pour tout $X, Y \in \mathcal{F}$. Si $(E, \mathcal{F})$ est un amas, alors on définit l'**amas bloquant** (en anglais *blocking clutter*) associé par

$$BL(E,\mathcal{F}) \;:=\; (E, \{X \subseteq E : X \cap Y \neq \emptyset \text{ pour tout } Y \in \mathcal{F}, \\ X \text{ minimal avec cette propriété}\}).$$

Pour un système d'indépendance donné $(E, \mathcal{F})$ et son dual $(E, \mathcal{F}^*)$, notons respectivement $\mathcal{B}$ et $\mathcal{B}^*$ les familles des bases, et $\mathcal{C}$ et $\mathcal{C}^*$ les familles des circuits de ces

systèmes. (Tout amas se décrit de ces deux façons sauf lorsque $\mathcal{F} = \emptyset$ ou $\mathcal{F} = \{\emptyset\}$.) Il découle immédiatement des définitions que $(E, \mathcal{B}^*) = BL(E, \mathcal{C})$ et $(E, \mathcal{C}^*) = BL(E, \mathcal{B})$. D'après la proposition 13.14, cela implique que $BL(BL(E, \mathcal{F})) = (E, \mathcal{F})$ pour tout amas $(E, \mathcal{F})$. Nous donnons quelques exemples d'amas $(E, \mathcal{F})$ et d'amas bloquants $(E, \mathcal{F}')$ associés. Dans chaque cas $E = E(G)$ pour un graphe G :

(1) $\mathcal{F}$ est l'ensemble des arbres couvrants, $\mathcal{F}'$ est l'ensemble des coupes minimales.

(2) $\mathcal{F}$ est l'ensemble des arborescences de racine r, $\mathcal{F}'$ est l'ensemble des coupes minimales issues de r.

(3) $\mathcal{F}$ est l'ensemble des chemins (resp. des chaînes si le graphe est non orienté) de s à t, $\mathcal{F}'$ est l'ensemble des coupes minimales séparant t de s (resp. s et t).

(4) $\mathcal{F}$ est l'ensemble des cycles d'un graphe non orienté, $\mathcal{F}'$ est l'ensemble des complémentaires des forêts maximales.

(5) $\mathcal{F}$ est l'ensemble des circuits d'un graphe orienté, $\mathcal{F}'$ est l'ensemble des arc-transversaux des circuits (en anglais *feedback edge sets*) minimaux (dans un graphe orienté, un arc-transversal des circuits est un ensemble d'arcs dont la suppression rend le graphe sans circuit).

(6) $\mathcal{F}$ est l'ensemble des ensembles minimaux d'arcs dont la contraction rend le graphe fortement orienté fortement connexe, $\mathcal{F}'$ est l'ensemble des coupes orientées minimales.

(7) $\mathcal{F}$ est l'ensemble des T-joints minimaux, $\mathcal{F}'$ est l'ensemble des T-coupes minimales.

Toutes ces relations peuvent être vérifiées facilement : (1) et (2) découlent directement des théorèmes 2.4 et 2.5, (3), (4) et (5) sont évidentes, (6) découle du corollaire 2.7 et (7) de la proposition 12.6.

Dans certains cas, l'amas bloquant fournit une caractérisation polyédrale du PROBLÈME DE MINIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE pour des fonctions coût non négatives :

Définition 13.17. *Soient $(E, \mathcal{F})$ un amas, $(E, \mathcal{F}')$ l'amas bloquant correspondant et P l'enveloppe convexe des vecteurs d'incidence des éléments de $\mathcal{F}$. On dit que $(E, \mathcal{F})$ vérifie la* **propriété flot-max/coupe-min** *si*

$$\left\{x + y : x \in P,\, y \in \mathbb{R}_+^E\right\} \;=\; \left\{x \in \mathbb{R}_+^E : \sum_{e \in B} x_e \geq 1 \text{ pour tout } B \in \mathcal{F}'\right\}.$$

Les exemples (2) et (7) de la liste précédente vérifient cette propriété (d'après les théorèmes 6.14 et 12.16), et également les exemples (3) et (6) (voir exercice 10). Le théorème suivant relie la formulation précédente du type couverture (*covering*) à une formulation du type empilement (*packing*) du problème dual et permet de déduire certains théorèmes min-max les uns des autres :

Théorème 13.18. (Fulkerson [1971], Lehman [1979]) *Soit $(E, \mathcal{F})$ un amas et $(E, \mathcal{F}')$ l'amas bloquant correspondant. Les affirmations suivantes sont équivalentes :*

(a) $(E, \mathcal{F})$ *vérifie la propriété flot-max/coupe-min.*

(b) $(E, \mathcal{F}')$ *vérifie la propriété flot-max/coupe-min.*

(c) $\min\{c(A) : A \in \mathcal{F}\} = \max\big\{ \mathbb{1}y : y \in \mathbb{R}_+^{\mathcal{F}'}, \sum_{B \in \mathcal{F}' : e \in B} y_B \leq c(e)$ *pour tout* $e \in E\big\}$ *pour tout* $c : E \to \mathbb{R}_+$.

Preuve. Comme $BL(E, \mathcal{F}') = BL(BL(E, \mathcal{F})) = (E, \mathcal{F})$, il suffit de prouver (a)⇒(c)⇒(b). L'implication (b)⇒(a) se déduit alors en échangeant les rôles de $\mathcal{F}$ et $\mathcal{F}'$.

(a)⇒(c) : d'après le corollaire 3.33, nous avons pour tout $c : E \to \mathbb{R}_+$

$$\min\{c(A) : A \in \mathcal{F}\} = \min\{cx : x \in P\} = \min\left\{c(x+y) : x \in P,\, y \in \mathbb{R}_+^E\right\},$$

où P est l'enveloppe convexe des vecteurs d'incidence des éléments de $\mathcal{F}$. Alors, d'après la propriété flot-max/coupe-min et le théorème de dualité 3.20, nous obtenons (c).

(c)⇒(b) : notons P' l'enveloppe convexe des vecteurs d'incidence des éléments de $\mathcal{F}'$. Nous devons montrer que

$$\left\{x + y : x \in P',\, y \in \mathbb{R}_+^E\right\} = \left\{ x \in \mathbb{R}_+^E : \sum_{e \in A} x_e \geq 1 \text{ pour tout } A \in \mathcal{F} \right\}.$$

Comme l'inclusion $\subseteq$ est évidente d'après la définition des amas bloquants, nous montrons uniquement l'autre inclusion. Considérons donc un vecteur $c \in \mathbb{R}_+^E$ tel que $\sum_{e \in A} c_e \geq 1$ pour tout $A \in \mathcal{F}$. D'après (c), nous avons

$$\begin{aligned} 1 &\leq \min\{c(A) : A \in \mathcal{F}\} \\ &= \max\left\{ \mathbb{1}y : y \in \mathbb{R}_+^{\mathcal{F}'}, \sum_{B \in \mathcal{F}' : e \in B} y_B \leq c(e) \text{ pour tout } e \in E \right\}, \end{aligned}$$

considérons donc un vecteur $y \in \mathbb{R}_+^{\mathcal{F}'}$ tel que $\mathbb{1}y = 1$ et $\sum_{B \in \mathcal{F}' : e \in B} y_B \leq c(e)$ pour tout $e \in E$. Alors $x_e := \sum_{B \in \mathcal{F}' : e \in B} y_B$ $(e \in E)$ définit un vecteur $x \in P'$ tel que $x \leq c$, ce qui prouve que $c \in \left\{x + y : x \in P',\, y \in \mathbb{R}_+^E\right\}$. □

Par exemple, ce théorème implique, de manière assez immédiate, le théorème flot-max/coupe-min 8.6 : soit (G, u, s, t) un réseau. D'après l'exercice 1 du chapitre 7, la longueur minimum d'un chemin de s à t dans (G, u) est égale au nombre maximum de coupes séparant t de s telles que chaque arc e soit contenu dans au plus $u(e)$ d'entre elles. Ainsi l'amas des chemins de s à t (exemple (3) de la liste précédente) vérifie la propriété flot-max/coupe-min, et donc l'amas bloquant correspondant la vérifie aussi. (c) appliqué à l'amas des coupes minimales de s à t implique alors le théorème flot-max/coupe-min.

Remarquons cependant que le théorème 13.18 ne garantit pas qu'il puisse exister un vecteur entier réalisant le maximum dans (c), même si c est entier. L'amas des T-joints pour $G = K_4$ et $T = V(G)$ montre qu'un tel vecteur n'existe pas en général.

13.4 Algorithme glouton

Considérons de nouveau un système d'indépendance $(E, \mathcal{F})$ et une fonction coût $c : E \to \mathbb{R}_+$. Nous considérons le PROBLÈME DE MAXIMISATION pour $(E, \mathcal{F}, c)$ et présentons deux «algorithmes gloutons». Nous n'avons pas à considérer des poids négatifs puisque des éléments de poids négatif ne peuvent apparaître dans une solution optimale.

Nous supposons que $(E, \mathcal{F})$ est donné par un oracle. Pour le premier algorithme, nous supposons simplement l'existence d'un **oracle d'indépendance**, c.-à-d. d'un oracle qui, pour un ensemble $F \subseteq E$, permet de décider si $F \in \mathcal{F}$ ou pas.

ALGORITHME GLOUTON-MEILLEUR-INSÉRÉ

Input Un système d'indépendance $(E, \mathcal{F})$, donné par un oracle d'indépendance. Des poids $c : E \to \mathbb{R}_+$.

Output Un ensemble $F \in \mathcal{F}$.

① Trier $E = \{e_1, e_2, \ldots, e_n\}$ de telle façon que $c(e_1) \geq c(e_2) \geq \cdots \geq c(e_n)$.

② $F := \emptyset$.

③ **For** $i := 1$ **to** n **do** : **If** $F \cup \{e_i\} \in \mathcal{F}$ **then** $F := F \cup \{e_i\}$.

Le deuxième algorithme nécessite un oracle plus compliqué. Pour un ensemble $F \subseteq E$ donné, cet oracle permet de décider si F contient une base. Un tel oracle sera appelé un **oracle de sur-ensemble de base** (en anglais *basis-superset oracle*).

ALGORITHME GLOUTON-PIRE-SORTI

Input Un système d'indépendance $(E, \mathcal{F})$, donné par un oracle de sur-ensemble de base. Poids $c : E \to \mathbb{R}_+$.

Output Une base F de $(E, \mathcal{F})$.

① Trier $E = \{e_1, e_2, \ldots, e_n\}$ de telle façon que $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_n)$.

② $F := E$.

③ **For** $i := 1$ **to** n **do** : **If** $F \setminus \{e_i\}$ contient une base **then** $F := F \setminus \{e_i\}$.

Avant d'analyser ces algorithmes, jetons un coup d'œil plus attentif aux oracles nécessaires. Une question intéressante est de savoir si de tels oracles sont polynomialement équivalents, c.-à-d. si l'un peut être simulé par un algorithme polynomial utilisant l'autre.

L'oracle d'indépendance et l'oracle de sur-ensemble de base ne semblent pas être polynomialement équivalents :

Si nous considérons le système d'indépendance associé au PVC (exemple (2) dans la liste du paragraphe 13.1), il est facile (et c'est le sujet de l'exercice 13) de décider si un ensemble d'arêtes est indépendant, c.-à-d. un sous-ensemble d'un

cycle hamiltonien (nous rappelons que nous travaillons avec un graphe complet). D'autre part, décider si un ensemble d'arêtes contient un cycle hamiltonien est un problème difficile (c'est un problème *NP*-complet ; voir théorème 15.25).

Inversement, pour le système d'indépendance associé au PROBLÈME DU PLUS COURT CHEMIN (exemple (3)), il est facile de décider si un ensemble d'arêtes contient un chemin de s à t. Mais on ne sait pas comment décider si un ensemble donné est indépendant (c.-à-d. un sous-ensemble d'un chemin de s à t) en temps polynomial. (Korte et Monma [1979] ont prouvé que ce problème est *NP*-complet.)

Dans le cas des matroïdes, les deux oracles sont polynomialement équivalents. Deux autres oracles équivalents sont l'**oracle de rang** et l'**oracle de fermeture**, qui fournissent respectivement le rang et la fermeture d'un sous-ensemble donné E (exercice 16).

Cependant, même pour les matroïdes, il existe d'autres oracles naturels qui ne sont pas polynomialement équivalents. Par exemple, l'oracle permettant de décider si un ensemble donné est une base est plus faible que l'oracle d'indépendance. L'oracle qui, pour un sous-ensemble $F \subseteq E$ donné, fournit le cardinal minimum d'un sous-ensemble dépendant de F est plus fort que l'oracle d'indépendance (Hausmann et Korte [1981]).

On peut formuler de manière analogue deux algorithmes gloutons pour le PROBLÈME DE MINIMISATION. Il est facile de voir que l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ appliqué au PROBLÈME DE MAXIMISATION pour $(E, \mathcal{F}, c)$ correspond à l'algorithme du type GLOUTON-PIRE-SORTI appliqué au PROBLÈME DE MINIMISATION pour $(E, \mathcal{F}^*, c)$: ajouter un élément à F dans l'approche GLOUTON-MEILLEUR-INSÉRÉ correspond à supprimer un élément de F dans celle GLOUTON-PIRE-SORTI. Remarquons que l'ALGORITHME DE KRUSKAL (voir paragraphe 6.1) est un algorithme du type GLOUTON-MEILLEUR-INSÉRÉ pour le PROBLÈME DE MINIMISATION dans un matroïde des cycles.

Le reste de ce paragraphe contient quelques résultats qui concernent l'évaluation de la qualité des solutions trouvées par les algorithmes gloutons.

Théorème 13.19. (Jenkyns [1976], Korte et Hausmann [1978]) *Soit $(E, \mathcal{F})$ un système d'indépendance. Pour $c : E \to \mathbb{R}_+$, notons $G(E, \mathcal{F}, c)$ le coût d'une solution fournie par l'algorithme du type* GLOUTON-MEILLEUR-INSÉRÉ *pour le* PROBLÈME DE MAXIMISATION*, et notons $\mathrm{OPT}(E, \mathcal{F}, c)$ le coût d'une solution optimale. Alors*

$$q(E, \mathcal{F}) \;\leq\; \frac{G(E, \mathcal{F}, c)}{\mathrm{OPT}(E, \mathcal{F}, c)} \;\leq\; 1$$

pour tout $c : E \to \mathbb{R}_+$. Il existe une fonction coût telle que la borne inférieure soit atteinte.

Preuve. Soit $E = \{e_1, e_2, \ldots, e_n\}$, $c : E \to \mathbb{R}_+$, tels que $c(e_1) \geq c(e_2) \geq \ldots \geq c(e_n)$. Soit G_n la solution fournie par l'algorithme GLOUTON-MEILLEUR-INSÉRÉ (lorsque E est trié ainsi) et soit O_n une solution optimale. Nous définissons $E_j := \{e_1, \ldots, e_j\}$, $G_j := G_n \cap E_j$ et $O_j := O_n \cap E_j$ ($j = 0, \ldots, n$). Posons $d_n := c(e_n)$ et $d_j := c(e_j) - c(e_{j+1})$ pour $j = 1, \ldots, n-1$.

Puisque $O_j \in \mathcal{F}$, nous avons $|O_j| \leq r(E_j)$. Puisque G_j est une base de E_j, nous avons $|G_j| \geq \rho(E_j)$. À l'aide de ces deux inégalités, nous pouvons conclure que

$$\begin{aligned} c(G_n) &= \sum_{j=1}^{n} (|G_j| - |G_{j-1}|)\, c(e_j) \\ &= \sum_{j=1}^{n} |G_j|\, d_j \\ &\geq \sum_{j=1}^{n} \rho(E_j)\, d_j \\ &\geq q(E,\mathcal{F}) \sum_{j=1}^{n} r(E_j)\, d_j \qquad (13.1) \\ &\geq q(E,\mathcal{F}) \sum_{j=1}^{n} |O_j|\, d_j \\ &= q(E,\mathcal{F}) \sum_{j=1}^{n} (|O_j| - |O_{j-1}|)\, c(e_j) \\ &= q(E,\mathcal{F})\, c(O_n). \end{aligned}$$

Finalement nous montrons que la borne inférieure est atteinte. Choisissons $F \subseteq E$ et des bases B_1, B_2 de F telles que

$$\frac{|B_1|}{|B_2|} = q(E,\mathcal{F}).$$

Définissons

$$c(e) := \begin{cases} 1 & \text{pour } e \in F \\ 0 & \text{pour } e \in E \setminus F \end{cases}$$

et trions $e_1, \ldots, e_n$ de telle façon que $c(e_1) \geq c(e_2) \geq \ldots \geq c(e_n)$ et $B_1 = \{e_1, \ldots, e_{|B_1|}\}$. Alors $G(E,\mathcal{F},c) = |B_1|$ et $\mathrm{OPT}(E,\mathcal{F},c) = |B_2|$, et la borne inférieure est atteinte. □

En particulier nous avons le théorème d'Edmonds-Rado :

Théorème 13.20. (Rado [1957], Edmonds [1971]) *Un système d'indépendance $(E,\mathcal{F})$ est un matroïde si et seulement si l'algorithme du type* GLOUTON-MEILLEUR-INSÉRÉ *fournit une solution optimale du* PROBLÈME DE MAXIMISATION *pour $(E,\mathcal{F},c)$ pour toutes les fonctions coûts $c : E \to \mathbb{R}_+$.*

Preuve. D'après le théorème 13.19, nous avons $q(E,\mathcal{F}) < 1$ si et seulement s'il existe une fonction coût $c : E \to \mathbb{R}_+$ pour laquelle l'algorithme GLOUTON-MEILLEUR-INSÉRÉ ne trouve pas une solution optimale. D'après la proposition 13.7, $q(E,\mathcal{F}) < 1$ si et seulement si $(E,\mathcal{F})$ n'est pas un matroïde. □

Cela est l'un des rares cas où l'on peut définir une structure à l'aide de son comportement algorithmique. Nous obtenons également une description polyédrale :

Théorème 13.21. (Edmonds [1970]) *Soit $(E,\mathcal{F})$ un matroïde et soit $r : 2^E \to \mathbb{Z}_+$ sa fonction rang. Alors le* **polytope associé au matroïde** *$(E,\mathcal{F})$, c.-à-d. l'enveloppe convexe des vecteurs d'incidence de tous les éléments de $\mathcal{F}$, est égale à*

$$\left\{ x \in \mathbb{R}^E : x \geq 0,\ \sum_{e\in A} x_e \leq r(A) \text{ pour tout } A \subseteq E \right\}.$$

Preuve. Évidemment, ce polytope contient tous les vecteurs d'incidence des ensembles indépendants. D'après le corollaire 3.32, il reste à montrer que tous les sommets de ce polytope sont entiers. D'après le théorème 5.13, cela équivaut à montrer que

$$\max \left\{ cx : x \geq 0,\ \sum_{e\in A} x_e \leq r(A) \text{ pour tout } A \subseteq E \right\} \tag{13.2}$$

a une solution optimale entière pour tout $c : E \to \mathbb{R}$. On peut toujours supposer que $c(e) \geq 0$ pour tout e, puisque pour $e \in E$ tel que $c(e) < 0$ toute solution optimale x de (13.2) vérifie $x_e = 0$.

Soit x une solution optimale de (13.2). Remplaçons dans (13.1), $|O_j|$ par $\sum_{e\in E_j} x_e$ $(j = 0,\dots,n)$. Nous obtenons $c(G_n) \geq \sum_{e\in E} c(e)x_e$. Alors l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ fournit une solution dont le vecteur d'incidence est une autre solution optimale de (13.2). □

Ce résultat, appliqué au cas d'un matroïde graphique, permet également de retrouver le théorème 6.12. Comme dans ce cas particulier, nous avons aussi, en général, la propriété de total dual intégralité. Une généralisation de ce résultat sera prouvée au paragraphe 14.2.

L'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ appliqué au PROBLÈME DE MAXIMISATION pour $(E,\mathcal{F},c)$ correspond, comme nous l'avons vu précédemment, à l'algorithme du type GLOUTON-PIRE-SORTI appliqué au PROBLÈME DE MINIMISATION pour $(E,\mathcal{F}^*,c)$. Cette observation suggère le théorème suivant, dual du théorème 13.19 :

Théorème 13.22. (Korte et Monma [1979]) *Soit $(E,\mathcal{F})$ un système d'indépendance. Pour $c : E \to \mathbb{R}_+$ notons $G(E,\mathcal{F},c)$ une solution fournie par l'algorithme* GLOUTON-PIRE-SORTI *appliqué au* PROBLÈME DE MINIMISATION. *Alors*

$$1 \ \leq\ \frac{G(E,\mathcal{F},c)}{\mathrm{OPT}(E,\mathcal{F},c)} \ \leq\ \max_{F\subseteq E} \frac{|F| - \rho^*(F)}{|F| - r^*(F)} \tag{13.3}$$

pour tout $c : E \to \mathbb{R}_+$, où ρ^* et r^* sont les fonctions rang du système d'indépendance dual $(E, \mathcal{F}^*)$. Il existe une fonction coût pour laquelle la borne supérieure est atteinte.

Preuve. Nous utilisons les mêmes notations que dans la preuve du théorème 13.19. Par construction, $G_j \cup (E \setminus E_j)$ contient une base de E, mais $(G_j \cup (E \setminus E_j)) \setminus \{e\}$ ne contient pas de base de E pour tout $e \in G_j$ $(j = 1, \ldots, n)$. En d'autres termes, $E_j \setminus G_j$ est une base de E_j relativement à $(E, \mathcal{F}^*)$, donc $|E_j| - |G_j| \geq \rho^*(E_j)$.

Puisque $O_n \subseteq E \setminus (E_j \setminus O_j)$ et que O_n est une base, $E_j \setminus O_j$ est un indépendant de $(E, \mathcal{F}^*)$, donc $|E_j| - |O_j| \leq r^*(E_j)$.

Nous en déduisons que

$$\begin{array}{rcll} |G_j| & \leq & |E_j| - \rho^*(E_j) & \text{et} \\ |O_j| & \geq & |E_j| - r^*(E_j). & \end{array}$$

Ensuite un calcul semblable à (13.1) fournit la borne supérieure. Afin de voir que cette borne est atteinte, considérons

$$c(e) := \begin{cases} 1 & \text{pour } e \in F \\ 0 & \text{pour } e \in E \setminus F \end{cases},$$

où $F \subseteq E$ est un ensemble tel que le maximum dans (13.3) est atteint. Soit B_1 une base de F relativement à $(E, \mathcal{F}^*)$, avec $|B_1| = \rho^*(F)$. Si on réordonne $e_1, \ldots, e_n$ de telle façon que $c(e_1) \geq c(e_2) \geq \ldots \geq c(e_n)$ et que $B_1 = \{e_1, \ldots, e_{|B_1|}\}$, on obtient $G(E, \mathcal{F}, c) = |F| - |B_1|$ et $\mathrm{OPT}(E, \mathcal{F}, c) = |F| - r^*(F)$. □

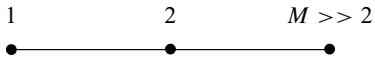


Figure 13.1.

Si on applique l'algorithme du type GLOUTON-PIRE-SORTI au PROBLÈME DE MAXIMISATION ou l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ au PROBLÈME DE MINIMISATION, il n'existe ni borne inférieure positive ni borne supérieure finie pour $\frac{G(E,\mathcal{F},c)}{\mathrm{OPT}(E,\mathcal{F},c)}$. Pour voir cela, on peut par exemple considérer le problème de la recherche d'une couverture minimale de poids maximum ou celui de la recherche d'un ensemble stable maximum de poids minimum dans le graphe simple de la figure 13.1.

Dans le cas des matroïdes, on peut utiliser indifféremment les algorithmes du type GLOUTON-MEILLEUR-INSÉRÉ ou GLOUTON-PIRE-SORTI. En effet, comme toutes les bases ont la même cardinalité, le PROBLÈME DE MINIMISATION appliqué à $(E, \mathcal{F}, c)$ est équivalent au PROBLÈME DE MAXIMISATION appliqué à $(E, \mathcal{F}, c')$, où $c'(e) := M - c(e)$ pour tout $e \in E$ et $M := 1 + \max\{c(e) : e \in E\}$.

Ainsi l'ALGORITHME DE KRUSKAL (paragraphe 6.1) permet de résoudre de façon optimale le PROBLÈME DE L'ARBRE COUVRANT MINIMUM.

Le théorème d'Edmonds-Rado 13.20 implique également la caractérisation suivante des solutions optimales de taille k du PROBLÈME DE MAXIMISATION.

Théorème 13.23. *Soient $(E, \mathcal{F})$ un matroïde, $c : E \to \mathbb{R}$, $k \in \mathbb{N}$ et $X \in \mathcal{F}$ tel que $|X| = k$. Alors $c(X) = \max\{c(Y) : Y \in \mathcal{F}, |Y| = k\}$ si et seulement si les deux conditions suivantes sont vérifiées :*

(a) *Pour tout $y \in E \setminus X$ tel que $X \cup \{y\} \notin \mathcal{F}$ et tout $x \in C(X, y)$ on a $c(x) \geq c(y)$;*

(b) *Pour tout $y \in E \setminus X$ tel que $X \cup \{y\} \in \mathcal{F}$ et tout $x \in X$ on a $c(x) \geq c(y)$.*

Preuve. La nécessité est évidente : si l'une des conditions est violée par un certain couple y, x, l'ensemble $X' := (X \cup \{y\}) \setminus \{x\} \in \mathcal{F}$ de taille k a un coût plus important que X.

Pour prouver la condition suffisante, considérons $\mathcal{F}' := \{F \in \mathcal{F} : |F| \leq k\}$ et $c'(e) := c(e) + M$ pour tout $e \in E$, où $M = \max\{|c(e)| : e \in E\}$. Réordonnons $E = \{e_1, \ldots, e_n\}$ de telle façon que $c'(e_1) \geq \cdots \geq c'(e_n)$ et que, pour tout i tel que $c'(e_i) = c'(e_{i+1})$ et $e_{i+1} \in X$, on ait $e_i \in X$ (autrement dit les éléments de X sont les premiers parmi ceux de poids égaux).

Soit X' la solution fournie par l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ pour l'instance $(E, \mathcal{F}', c')$. Comme $(E, \mathcal{F}')$ est un matroïde, le théorème d'Edmonds-Rado 13.20 implique :

$$\begin{aligned} c(X') + kM &= c'(X') = \max\{c'(Y) : Y \in \mathcal{F}'\} \\ &= \max\{c(Y) : Y \in \mathcal{F}, |Y| = k\} + kM. \end{aligned}$$

Nous terminons la preuve en montrant que $X = X'$. On sait que $|X| = k = |X'|$. Supposons donc $X \neq X'$ et considérons $e_i \in X' \setminus X$ avec i minimum. Alors $X \cap \{e_1, \ldots, e_{i-1}\} = X' \cap \{e_1, \ldots, e_{i-1}\}$. Si maintenant $X \cup \{e_i\} \notin \mathcal{F}$, alors (a) implique $C(X, e_i) \subseteq X'$, une contradiction. Si $X \cup \{e_i\} \in \mathcal{F}$, alors (b) implique $X \subseteq X'$, ce qui est également impossible. □

Nous aurons besoin de ce théorème au paragraphe 13.7. Le cas particulier où $(E, \mathcal{F})$ est un matroïde graphique et où $k = r(E)$ correspond au théorème 6.2.

13.5 Intersection de matroïdes

Définition 13.24. *Soient deux systèmes d'indépendance donnés $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$, on définit leur* **intersection** *par $(E, \mathcal{F}_1 \cap \mathcal{F}_2)$.*

L'intersection d'un nombre fini de systèmes d'indépendance est défini de manière analogue. Il est clair que l'intersection de systèmes d'indépendance est encore un système d'indépendance.

Proposition 13.25. *Tout système d'indépendance $(E, \mathcal{F})$ est l'intersection d'un nombre fini de matroïdes.*

Preuve. Chaque circuit C de $(E, \mathcal{F})$ définit un matroïde $(E, \{F \subseteq E : C \setminus F \neq \emptyset\})$ d'après le théorème 13.12. L'intersection de tous ces matroïdes correspond à $(E, \mathcal{F})$. □

Comme l'intersection de matroïdes n'est généralement pas un matroïde, on ne peut espérer obtenir un ensemble indépendant optimal commun à plusieurs matroïdes à l'aide d'un algorithme glouton. Cependant, le résultat suivant et le théorème 13.19, procurent une borne pour la solution fournie par l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ :

Proposition 13.26. *Si $(E, \mathcal{F})$ est l'intersection de p matroïdes, $q(E, \mathcal{F}) \geq \frac{1}{p}$.*

Preuve. D'après le théorème 13.12(b), $X \cup \{e\}$ contient au plus p circuits pour tout $X \in \mathcal{F}$ et $e \in E$. L'affirmation découle maintenant du théorème 13.8. □

Les systèmes d'indépendance qui correspondent à l'intersection de deux matroïdes sont particulièrement intéressants. Un exemple classique est le problème du couplage dans un graphe biparti $G = (A \dot{\cup} B, E)$. Si $\mathcal{F} := \{F \subseteq E : F \text{ est un couplage dans } G\}$, alors $(E, \mathcal{F})$ est l'intersection de deux matroïdes. En effet, considérons

$$\begin{aligned} \mathcal{F}_1 &:= \{F \subseteq E : |\delta_F(x)| \leq 1 \text{ pour tout } x \in A\} \quad \text{et} \\ \mathcal{F}_2 &:= \{F \subseteq E : |\delta_F(x)| \leq 1 \text{ pour tout } x \in B\}. \end{aligned}$$

$(E, \mathcal{F}_1), (E, \mathcal{F}_2)$ sont des matroïdes d'après la proposition 13.4(d). Évidemment, $\mathcal{F} = \mathcal{F}_1 \cap \mathcal{F}_2$.

Un deuxième exemple est le système d'indépendance correspondant à toutes les ramifications d'un graphe orienté G (exemple 8 de la liste au début du paragraphe 13.1). Ici le premier matroïde contient tous les ensembles d'arcs tels que chaque sommet ait au plus un arc entrant (voir proposition 13.4(e)), tandis que le second matroïde est le matroïde des cycles $\mathcal{M}(G)$ du graphe non orienté associé.

Nous allons maintenant décrire l'algorithme d'Edmonds pour le problème suivant :

PROBLÈME DE L'INTERSECTION DE MATROÏDES	
Instance	Deux matroïdes $(E, \mathcal{F}_1), (E, \mathcal{F}_2)$, donnés par des oracles d'indépendance.
Tâche	Trouver un ensemble $F \in \mathcal{F}_1 \cap \mathcal{F}_2$ tel que $\lvert F \rvert$ soit maximum.

Commençons par le lemme suivant. Rappelons que, pour $X \in \mathcal{F}$ et $e \in E$, on note $C(X, e)$ l'unique circuit de $X \cup \{e\}$ si $X \cup \{e\} \notin \mathcal{F}$, et $C(X, e) = \emptyset$ sinon.

Lemme 13.27. (Frank [1981]) *Soient $(E, \mathcal{F})$ un matroïde et $X \in \mathcal{F}$. Soient $x_1, \ldots, x_s \in X$ et $y_1, \ldots, y_s \notin X$ tels que :*

(a) $x_k \in C(X, y_k)$ *pour* $k = 1, \ldots, s$ *et*

(b) $x_j \notin C(X, y_k)$ *pour* $1 \leq j < k \leq s$.

Alors $(X \setminus \{x_1, \ldots, x_s\}) \cup \{y_1, \ldots, y_s\} \in \mathcal{F}$.

Preuve. Soit $X_r := (X \setminus \{x_1, \ldots, x_r\}) \cup \{y_1, \ldots, y_r\}$. Nous allons montrer que $X_r \in \mathcal{F}$ pour tout r par induction. Pour $r = 0$ le résultat est évident. Supposons alors que $X_{r-1} \in \mathcal{F}$ pour $r \in \{1, \ldots, s\}$. Si $X_{r-1} \cup \{y_r\} \in \mathcal{F}$ alors on a immédiatement $X_r \in \mathcal{F}$. Sinon $X_{r-1} \cup \{y_r\}$ contient un circuit C unique (d'après le théorème 13.12(b)). Comme $C(X, y_r) \subseteq X_{r-1} \cup \{y_r\}$ (d'après (b)), on doit avoir $C = C(X, y_r)$. Mais alors, d'après (a), $x_r \in C(X, y_r) = C$, donc $X_r = (X_{r-1} \cup \{y_r\}) \setminus \{x_r\} \in \mathcal{F}$. □

L'idée de base de l'ALGORITHME DE L'INTERSECTION DE MATROÏDES D'EDMONDS est la suivante. Partant de $X = \emptyset$, on augmente X d'un élément à chaque itération. Comme on ne peut généralement pas espérer trouver directement un élément e tel que $X \cup \{e\} \in \mathcal{F}_1 \cap \mathcal{F}_2$, nous rechercherons des «chemins alternés». Pour rendre cela plus commode, nous définissons un graphe auxiliaire. Nous généralisons les ensembles du type $C(X, e)$ à $(E, \mathcal{F}_i)$ en écrivant $C_i(X, e)$ $(i = 1, 2)$.

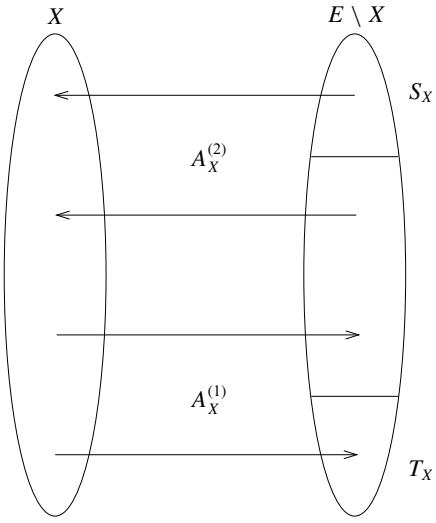


Figure 13.2.

Soit un ensemble donné $X \in \mathcal{F}_1 \cap \mathcal{F}_2$, nous définissons un graphe auxiliaire orienté G_X par

$$\begin{aligned} A_X^{(1)} &:= \{\, (x, y) : y \in E \setminus X,\ x \in C_1(X, y) \setminus \{y\} \,\}, \\ A_X^{(2)} &:= \{\, (y, x) : y \in E \setminus X,\ x \in C_2(X, y) \setminus \{y\} \,\}, \\ G_X &:= (E, A_X^{(1)} \cup A_X^{(2)}). \end{aligned}$$

Nous posons

$$\begin{aligned} S_X &:= \{y \in E \setminus X : X \cup \{y\} \in \mathcal{F}_1\}, \\ T_X &:= \{y \in E \setminus X : X \cup \{y\} \in \mathcal{F}_2\} \end{aligned}$$

(voir figure 13.2) et recherchons un plus court chemin de S_X à T_X. Un tel ensemble nous permettra d'augmenter l'ensemble X. (Si $S_X \cap T_X \neq \emptyset$, on a un chemin de longueur zéro et l'on peut augmenter X par un élément de $S_X \cap T_X$.)

Lemme 13.28. *Soit $X \in \mathcal{F}_1 \cap \mathcal{F}_2$. Soient $y_0, x_1, y_1, \ldots, x_s, y_s$ les sommets d'un plus court chemin de y_0 à y_s de G_X (pris dans cet ordre), tels que $y_0 \in S_X$ et $y_s \in T_X$. Alors*

$$X' := (X \cup \{y_0, \ldots, y_s\}) \setminus \{x_1, \ldots, x_s\} \in \mathcal{F}_1 \cap \mathcal{F}_2.$$

Preuve. Montrons d'abord que $X \cup \{y_0\}$, $x_1, \ldots, x_s$ et $y_1, \ldots, y_s$ satisfont les conditions du lemme 13.27 appliqué à $\mathcal{F}_1$. Remarquons que $X \cup \{y_0\} \in \mathcal{F}_1$, car $y_0 \in S_X$. (a) est vérifié, car $(x_j, y_j) \in A_X^{(1)}$ pour tout j, et (b) est également vérifié sinon le chemin pourrait être raccourci. Nous en concluons que $X' \in \mathcal{F}_1$.

Ensuite, nous montrons que $X \cup \{y_s\}$, $x_s, x_{s-1}, \ldots, x_1$ et $y_{s-1}, \ldots, y_1, y_0$ satisfont les conditions du lemme 13.27 appliqué à $\mathcal{F}_2$. Remarquons que $X \cup \{y_s\} \in \mathcal{F}_2$, car $y_s \in T_X$. (a) est vérifié, car $(y_{j-1}, x_j) \in A_X^{(2)}$ pour tout j, et (b) est également vérifié sinon le chemin pourrait être raccourci. Nous en concluons que $X' \in \mathcal{F}_2$. □

Nous allons maintenant prouver que, s'il n'existe pas de chemin de S_X à T_X dans G_X, alors X est déjà maximum. Nous avons besoin du résultat simple suivant :

Proposition 13.29. *Soient $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$ deux matroïdes de fonctions rang r_1 et r_2. Alors pour tout $F \in \mathcal{F}_1 \cap \mathcal{F}_2$ et tout $Q \subseteq E$ on a*

$$|F| \leq r_1(Q) + r_2(E \setminus Q).$$

Preuve. $F \cap Q \in \mathcal{F}_1$ implique $|F \cap Q| \leq r_1(Q)$. De la même façon $F \setminus Q \in \mathcal{F}_2$ implique $|F \setminus Q| \leq r_2(E \setminus Q)$. En ajoutant les deux inégalités, on obtient le résultat. □

Lemme 13.30. *$X \in \mathcal{F}_1 \cap \mathcal{F}_2$ est maximum si et seulement s'il n'existe pas de chemin de S_X à T_X dans G_X.*

Preuve. S'il existe un chemin de S_X à T_X, il existe un plus court chemin de S_X à T_X. Nous appliquons le lemme 13.28 et nous obtenons alors un ensemble $X' \in \mathcal{F}_1 \cap \mathcal{F}_2$ de plus grand cardinal que X.

Pour l'autre sens, soit R l'ensemble des sommets que l'on peut atteindre depuis S_X dans G_X (voir figure 13.3). On a $R \cap T_X = \emptyset$. Soient r_1 et r_2 les fonctions rang de $\mathcal{F}_1$ et $\mathcal{F}_2$, respectivement.

Nous affirmons que $r_2(R) = |X \cap R|$. Si ce n'était pas le cas, il existerait un $y \in R \setminus X$ tel que $(X \cap R) \cup \{y\} \in \mathcal{F}_2$. Comme $X \cup \{y\} \notin \mathcal{F}_2$ (car $y \notin T_X$), le

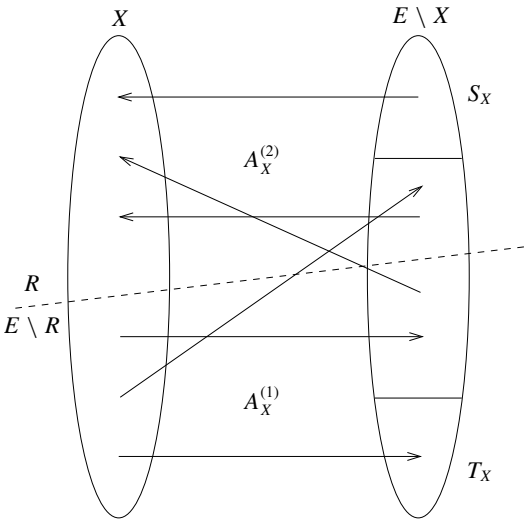


Figure 13.3.

circuit $C_2(X, y)$ doit contenir un élément $x \in X \setminus R$. Mais alors $(y, x) \in A_X^{(2)}$ et donc il existe un arc sortant de R. Cela contredit la définition de R.

Prouvons maintenant que $r_1(E \setminus R) = |X \setminus R|$. Si ce n'était pas le cas, il existerait un $y \in (E \setminus R) \setminus X$ tel que $(X \setminus R) \cup \{y\} \in \mathcal{F}_1$. Comme $X \cup \{y\} \notin \mathcal{F}_1$ (car $y \notin S_X$), le circuit $C_1(X, y)$ doit contenir un élément $x \in X \cap R$. Mais alors $(x, y) \in A_X^{(1)}$ et donc il existe un arc sortant de R. Cela contredit la définition de R.

En réunissant ces résultats, on obtient $|X| = r_2(R) + r_1(E \setminus R)$. D'après la proposition 13.29, cela implique l'optimalité. □

Le dernier paragraphe de cette preuve fournit l'égalité min-max suivante :

Théorème 13.31. (Edmonds [1970]) *Soient $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$ deux matroïdes de fonctions rang r_1 et r_2. Alors*

$$\max\{|X| : X \in \mathcal{F}_1 \cap \mathcal{F}_2\} = \min\{r_1(Q) + r_2(E \setminus Q) : Q \subseteq E\}.$$

□

Nous pouvons maintenant donner une description détaillée de l'algorithme.

ALGORITHME DE L'INTERSECTION DE MATROÏDES D'EDMONDS	
Input	Deux matroïdes $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$, donnés par des oracles d'indépendance.
Output	Un ensemble $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ de cardinalité maximale.

① Poser $X := \emptyset$.

② **For** tout $y \in E \setminus X$ et $i \in \{1,2\}$ **do** : Calculer
$C_i(X,y) := \{x \in X \cup \{y\} : X \cup \{y\} \notin \mathcal{F}_i,\ (X \cup \{y\}) \setminus \{x\} \in \mathcal{F}_i\}$.

③ Calculer S_X, T_X, et G_X comme définis précédemment.

④ Appliquer BFS pour trouver un plus court chemin de S_X à T_X P dans G_X. **If** il n'en existe pas **then stop**.

⑤ Poser $X := X \triangle V(P)$ et **go to** ②.

Théorème 13.32. *L'*ALGORITHME DE L'INTERSECTION DE MATROÏDES D'EDMONDS *résout correctement le* PROBLÈME DE L'INTERSECTION DE MATROÏDES *en $O(|E|^3\theta)$, où θ est la complexité maximale des deux oracles d'indépendance.*

Preuve. La validité de l'algorithme est impliquée par les lemmes 13.28 et 13.30. ② et ③ peuvent être réalisés en $O(|E|^2\theta)$, ④ en temps $O(|E|)$. Comme il y a au plus $|E|$ augmentations, la complexité globale est en $O(|E|^3\theta)$. □

Des algorithmes de l'intersection de matroïdes plus rapides ont été proposés par Cunningham [1986] et Gabow et Xu [1996]. Remarquons que le problème de la recherche d'un ensemble de cardinal maximum dans l'intersection de trois matroïdes est un problème *NP*-difficile ; voir l'exercice 14(c) du chapitre 15.

13.6 Partition de matroïdes

Au lieu de l'intersection de matroïdes, nous considérons maintenant l'union de matroïdes qui est définie de la manière suivante :

Définition 13.33. *Soient k matroïdes : $(E,\mathcal{F}_1),\dots,(E,\mathcal{F}_k)$. Un ensemble $X \subseteq E$ est dit* **ensemble partitionnable** *s'il existe une partition $X = X_1 \dot\cup \cdots \dot\cup X_k$ telle que $X_i \in \mathcal{F}_i$ pour $i = 1,\dots,k$. Soit $\mathcal{F}$ la famille des sous-ensembles partitionnables de E. Alors $(E,\mathcal{F})$ est appelé l'***union** *ou la* **somme** *de $(E,\mathcal{F}_1),\dots,(E,\mathcal{F}_k)$.*

Nous prouverons que l'union de matroïdes est encore un matroïde. De plus, nous allons résoudre le problème suivant en utilisant l'intersection de matroïdes :

PROBLÈME DE LA PARTITION DE MATROÏDES	
Instance	Un nombre $k \in \mathbb{N}$, k matroïdes $(E,\mathcal{F}_1),\dots,(E,\mathcal{F}_k)$, donnés par des oracles d'indépendance.
Tâche	Trouver un ensemble partitionnable $X \subseteq E$ de cardinalité maximale.

Le théorème principal concernant la partition de matroïdes est le suivant :

Théorème 13.34. (Nash-Williams [1967]) *Soient $(E,\mathcal{F}_1),\dots,(E,\mathcal{F}_k)$ des matroïdes de fonctions rang $r_1,\dots,r_k$, et soit $(E,\mathcal{F})$ leur union. Alors $(E,\mathcal{F})$ est un matroïde, et sa fonction rang r est définie par*

$$r(X) = \min_{A \subseteq X} \left(|X \setminus A| + \sum_{i=1}^{k} r_i(A) \right).$$

Preuve. $(E, \mathcal{F})$ est évidemment un système d'indépendance. Soit $X \subseteq E$. Nous prouvons d'abord que $r(X) = \min_{A \subseteq X} \left(|X \setminus A| + \sum_{i=1}^{k} r_i(A) \right)$.

Pour tout $Y \subseteq X$ tel que Y soit partitionnable, c.-à-d. $Y = Y_1 \dot\cup \cdots \dot\cup Y_k$ avec $Y_i \in \mathcal{F}_i$ $(i = 1, \ldots, k)$, et pour tout $A \subseteq X$ on a

$$|Y| \;=\; |Y \setminus A| + |Y \cap A| \;\leq\; |X \setminus A| + \sum_{i=1}^{k} |Y_i \cap A| \;\leq\; |X \setminus A| + \sum_{i=1}^{k} r_i(A),$$

donc $r(X) \leq \min_{A \subseteq X} \left(|X \setminus A| + \sum_{i=1}^{k} r_i(A) \right)$.

D'autre part, soit $X' := X \times \{1, \ldots, k\}$. Nous définissons deux matroïdes sur X'. Pour $Q \subseteq X'$ et $i \in \{1, \ldots, k\}$, nous notons $Q_i := \{e \in X : (e, i) \in Q\}$. Considérons

$$\mathcal{I}_1 \;:=\; \{Q \subseteq X' : Q_i \in \mathcal{F}_i \text{ pour tout } i = 1, \ldots, k\}$$

et

$$\mathcal{I}_2 \;:=\; \{Q \subseteq X' : Q_i \cap Q_j = \emptyset \text{ pour tout } i \neq j\}.$$

Évidemment, $(X', \mathcal{I}_1)$ et $(X', \mathcal{I}_2)$ sont tous deux des matroïdes, et leurs fonctions rang sont définies par $s_1(Q) := \sum_{i=1}^{k} r_i(Q_i)$ et $s_2(Q) := \left| \bigcup_{i=1}^{k} Q_i \right|$ pour $Q \subseteq X'$.

La famille des sous-ensembles partitionnables de X peut être maintenant décrite par

$$\{A \subseteq X : \text{il existe une fonction } f : A \to \{1, \ldots, k\} \text{ telle que } \{(e, f(e)) : e \in A\} \in \mathcal{I}_1 \cap \mathcal{I}_2\}.$$

La cardinalité maximale d'un ensemble partitionnable est donc égale à la cardinalité maximale d'un ensemble indépendant commun à $\mathcal{I}_1$ et $\mathcal{I}_2$. D'après le théorème 13.31, cette cardinalité maximale est égale à

$$\min \{s_1(Q) + s_2(X' \setminus Q) : Q \subseteq X'\}.$$

Si $Q \subseteq X'$ atteint ce minimum, alors pour $A := Q_1 \cap \cdots \cap Q_k$ on a

$$r(X) \;=\; s_1(Q) + s_2(X' \setminus Q) \;=\; \sum_{i=1}^{k} r_i(Q_i) + \left| X \setminus \bigcap_{i=1}^{k} Q_i \right| \;\geq\; \sum_{i=1}^{k} r_i(A) + |X \setminus A|.$$

On a donc trouvé un ensemble $A \subseteq X$ tel que $\sum_{i=1}^{k} r_i(A) + |X \setminus A| \leq r(X)$.

Nous avons ainsi prouvé la formule de la fonction rang r. Il nous reste à montrer que r est sous-modulaire. D'après le théorème 13.10, cela implique alors que $(E, \mathcal{F})$ est un matroïde. Pour prouver la sous-modularité, considérons $X, Y \subseteq E$, et $A \subseteq$

$X, B \subseteq Y$ tels que $r(X) = |X \setminus A| + \sum_{i=1}^{k} r_i(A)$ et $r(Y) = |Y \setminus B| + \sum_{i=1}^{k} r_i(B)$. Alors

$$\begin{aligned} & r(X) + r(Y) \\ = \; & |X \setminus A| + |Y \setminus B| + \sum_{i=1}^{k} (r_i(A) + r_i(B)) \\ \geq \; & |(X \cup Y) \setminus (A \cup B)| + |(X \cap Y) \setminus (A \cap B)| + \sum_{i=1}^{k} (r_i(A \cup B) + r_i(A \cap B)) \\ \geq \; & r(X \cup Y) + r(X \cap Y). \end{aligned}$$

□

La construction de la preuve précédente (Edmonds [1970]) réduit le PROBLÈME DE LA PARTITION DE MATROÏDES au PROBLÈME DE L'INTERSECTION DE MATROÏDES. Une réduction dans le sens inverse est également possible (exercice 20) : ainsi les deux problèmes peuvent être considérés comme équivalents.

Remarquons que l'on peut trouver efficacement un ensemble indépendant de cardinal maximum dans l'union d'un nombre quelconque de matroïdes, alors que l'on ne sait pas traiter efficacement le problème de l'intersection de plus de deux matroïdes.

13.7 Intersection de matroïdes avec poids

Nous considérons maintenant une généralisation de l'algorithme précédent au cas pondéré.

PROBLÈME DE L'INTERSECTION DE MATROÏDES AVEC POIDS	
Instance	Deux matroïdes $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$, donnés par des oracles d'indépendance. Poids $c : E \to \mathbb{R}$.
Tâche	Trouver un ensemble $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ dont le poids $c(X)$ est maximum.

Nous décrirons pour résoudre ce problème un algorithme primal-dual dû à Frank [1981]. Il généralise l'algorithme de l'intersection de matroïdes d'EDMONDS. Nous partons de nouveau de $X := X_0 = \emptyset$ et augmentons la cardinalité d'un à chaque itération. Nous obtenons des ensembles $X_0, \ldots, X_m \in \mathcal{F}_1 \cap \mathcal{F}_2$ tels que $|X_k| = k$ $(k = 0, \ldots, m)$ et $m = \max\{|X| : X \in \mathcal{F}_1 \cap \mathcal{F}_2\}$. Chaque X_k sera optimal, c.-à-d.

$$c(X_k) \; = \; \max\{c(X) : X \in \mathcal{F}_1 \cap \mathcal{F}_2, \, |X| = k\}. \tag{13.4}$$

Ainsi, à la fin, il ne nous reste plus qu'à choisir l'ensemble optimal dans l'ensemble $X_0, \ldots, X_m$.

L'idée principale est de diviser la fonction poids. À chaque étape on a deux fonctions $c_1, c_2 : E \to \mathbb{R}$ telles que $c_1(e) + c_2(e) = c(e)$ pour tout $e \in E$. Pour chaque k nous garantirons

$$c_i(X_k) \;=\; \max\{c_i(X) : X \in \mathcal{F}_i,\, |X| = k\} \qquad (i = 1, 2). \tag{13.5}$$

Cette condition implique évidemment (13.4). Pour obtenir (13.5) nous utilisons le critère d'optimalité du théorème 13.23. Au lieu de G_X, S_X et T_X, on considère seulement un sous-graphe $\bar{G}$ et des sous-ensembles $\bar{S}$, $\bar{T}$.

ALGORITHME DE L'INTERSECTION DE MATROÏDES AVEC POIDS

Input Deux matroïdes $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$, donnés par des oracles d'indépendance.
Poids $c : E \to \mathbb{R}$.

Output Un ensemble $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ de poids maximum.

① Poser $k := 0$ et $X_0 := \emptyset$. Poser $c_1(e) := c(e)$ et $c_2(e) := 0$ pour tout $e \in E$.

② **For** chaque $y \in E \setminus X_k$ et $i \in \{1, 2\}$ **do** : Calculer
$C_i(X_k, y) := \{x \in X_k \cup \{y\} : X_k \cup \{y\} \notin \mathcal{F}_i,\, (X_k \cup \{y\}) \setminus \{x\} \in \mathcal{F}_i\}$.

③ Calculer

$$\begin{aligned}
A^{(1)} &:= \{\,(x, y) : y \in E \setminus X_k,\; x \in C_1(X_k, y) \setminus \{y\}\,\},\\
A^{(2)} &:= \{\,(y, x) : y \in E \setminus X_k,\; x \in C_2(X_k, y) \setminus \{y\}\,\},\\
S &:= \{\,y \in E \setminus X_k : X_k \cup \{y\} \in \mathcal{F}_1\,\},\\
T &:= \{\,y \in E \setminus X_k : X_k \cup \{y\} \in \mathcal{F}_2\,\}.
\end{aligned}$$

④ Calculer

$$\begin{aligned}
m_1 &:= \max\{c_1(y) : y \in S\}\\
m_2 &:= \max\{c_2(y) : y \in T\}\\
\bar{S} &:= \{\,y \in S : c_1(y) = m_1\,\}\\
\bar{T} &:= \{\,y \in T : c_2(y) = m_2\,\}\\
\bar{A}^{(1)} &:= \{\,(x, y) \in A^{(1)} : c_1(x) = c_1(y)\,\},\\
\bar{A}^{(2)} &:= \{\,(y, x) \in A^{(2)} : c_2(x) = c_2(y)\,\},\\
\bar{G} &:= (E, \bar{A}^{(1)} \cup \bar{A}^{(2)}).
\end{aligned}$$

⑤ Appliquer BFS pour calculer l'ensemble R de sommets que l'on peut atteindre depuis $\bar{S}$ dans $\bar{G}$.

⑥ **If** $R \cap \bar{T} \neq \emptyset$ **then** : trouver un chemin de $\bar{S}$ à $\bar{T}$ P dans $\bar{G}$ avec un nombre minimum d'arcs, poser $X_{k+1} := X_k \triangle V(P)$ et $k := k + 1$ et **go to** ②.

⑦ Calculer

$$\begin{aligned}
\varepsilon_1 &:= \min\{c_1(x) - c_1(y) : (x,y) \in A^{(1)} \cap \delta^+(R)\};\\
\varepsilon_2 &:= \min\{c_2(x) - c_2(y) : (y,x) \in A^{(2)} \cap \delta^+(R)\};\\
\varepsilon_3 &:= \min\{m_1 - c_1(y) : y \in S \setminus R\};\\
\varepsilon_4 &:= \min\{m_2 - c_2(y) : y \in T \cap R\};\\
\varepsilon &:= \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\}
\end{aligned}$$

(où $\min \emptyset := \infty$).

⑧ **If** $\varepsilon < \infty$ **then** :
Poser $c_1(x) := c_1(x) - \varepsilon$ et $c_2(x) := c_2(x) + \varepsilon$ pour tout $x \in R$. **Go to** ④.
If $\varepsilon = \infty$ **then** :
Parmi $X_0, X_1, \ldots, X_k$, soit X celui de poids maximum. **Stop**.

Voir Edmonds [1979] et Lawler [1976] pour des versions plus anciennes de cet algorithme.

Théorème 13.35. (Frank [1981]) *L'*ALGORITHME DE L'INTERSECTION DE MATROÏDES AVEC POIDS *résout correctement le* PROBLÈME DE L'INTERSECTION DE MATROÏDES AVEC POIDS *en temps $O(|E|^4 + |E|^3\theta)$, où θ est la complexité maximale des deux oracles d'indépendance.*

Preuve. Soit m la valeur finale de k. L'algorithme calcule successivement les ensembles $X_0, X_1, \ldots, X_m$. Prouvons que $X_k \in \mathcal{F}_1 \cap \mathcal{F}_2$ pour $k = 0, \ldots, m$, par induction sur k. Cela est évident pour $k = 0$. Si nous travaillons avec $X_k \in \mathcal{F}_1 \cap \mathcal{F}_2$ pour un certain k, $\bar{G}$ est un sous-graphe de $(E, A^{(1)} \cup A^{(2)}) = G_{X_k}$. Donc, si un chemin P est trouvé à l'étape ⑤, le lemme 13.28 assure que $X_{k+1} \in \mathcal{F}_1 \cap \mathcal{F}_2$.

Lorsque l'algorithme s'arrête, on a $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \infty$, on ne peut donc pas atteindre T depuis S dans G_{X_m}. Alors, d'après le lemme 13.30, $m = |X_m| = \max\{|X| : X \in \mathcal{F}_1 \cap \mathcal{F}_2\}$.

Pour prouver la validité de l'algorithme, nous montrons que pour $k = 0, \ldots, m$, $c(X_k) = \max\{c(X) : X \in \mathcal{F}_1 \cap \mathcal{F}_2, |X| = k\}$. Comme nous avons toujours $c = c_1 + c_2$, il suffit de prouver, qu'à chaque étape de l'algorithme, (13.5) est vérifié. Cela est évidemment vrai lorsque l'algorithme démarre (pour $k = 0$). Nous allons montrer que (13.5) n'est jamais violé en utilisant le théorème 13.23.

Lorsque nous posons $X_{k+1} := X_k \triangle V(P)$ à l'étape ⑥, nous devons nous assurer que (13.5) est vérifié. Soit P un chemin de s à t, $s \in \bar{S}$, $t \in \bar{T}$. Par définition de $\bar{G}$, on a $c_1(X_{k+1}) = c_1(X_k) + c_1(s)$ et $c_2(X_{k+1}) = c_2(X_k) + c_2(t)$. Comme X_k satisfait (13.5), les conditions (a) et (b) du théorème 13.23 appliqué à $\mathcal{F}_1$ et $\mathcal{F}_2$ doivent être vérifiées pour X_k.

D'après la définition de $\bar{S}$, ces deux conditions sont encore vérifiées pour $X_k \cup \{s\}$ et $\mathcal{F}_1$. Ainsi $c_1(X_{k+1}) = c_1(X_k \cup \{s\}) = \max\{c_1(Y) : Y \in \mathcal{F}_1, |Y| = k+1\}$. De même, d'après la définition de $\bar{T}$, les conditions (a) et (b) du théorème 13.23 sont

encore vérifiées pour $X_k \cup \{t\}$ et $\mathcal{F}_2$, ce qui implique $c_2(X_{k+1}) = c_2(X_k \cup \{t\}) = \max\{c_2(Y) : Y \in \mathcal{F}_2, |Y| = k+1\}$. Autrement dit, (13.5) est bien vérifié pour X_{k+1}.

Supposons maintenant que nous modifions c_1 et c_2 à l'étape ⑧. Nous montrons d'abord que $\varepsilon > 0$. D'après (13.5) et le théorème 13.23, on a $c_1(x) \geq c_1(y)$ pour tout $y \in E \setminus X_k$ et $x \in C_1(X_k, y) \setminus \{y\}$. Donc, pour tout $(x, y) \in A^{(1)}$, on a $c_1(x) \geq c_1(y)$. De plus, par définition de R, il n'y a pas d'arc $(x, y) \in \delta^+(R)$ qui puisse appartenir à $\bar{A}^{(1)}$. Cela implique $\varepsilon_1 > 0$.

$\varepsilon_2 > 0$ se prouve de manière similaire. $m_1 \geq c_1(y)$ est vérifié pour tout $y \in S$. Si en plus $y \notin R$ alors $y \notin \bar{S}$, donc $m_1 > c_1(y)$. Ainsi $\varepsilon_3 > 0$. De la même façon, $\varepsilon_4 > 0$ (en utilisant $\bar{T} \cap R = \emptyset$). On en déduit que $\varepsilon > 0$.

Nous pouvons maintenant prouver que l'étape ⑧ préserve (13.5). Soit c_1' le poids modifié de c_1, c.-à-d.

$$c_1'(x) := \begin{cases} c_1(x) - \varepsilon & \text{si } x \in R \\ c_1(x) & \text{si } x \notin R \end{cases}.$$

Nous prouvons que X_k et c_1' vérifient les conditions du théorème 13.23 appliqué à $\mathcal{F}_1$.

Afin de prouver (a), considérons $y \in E \setminus X_k$ et $x \in C_1(X_k, y) \setminus \{y\}$. Supposons $c_1'(x) < c_1'(y)$. Comme $c_1(x) \geq c_1(y)$ et $\varepsilon > 0$, on doit avoir $x \in R$ et $y \notin R$. Comme $(x, y) \in A^{(1)}$, on a $\varepsilon \leq \varepsilon_1 \leq c_1(x) - c_1(y) = (c_1'(x) + \varepsilon) - c_1'(y)$ et on obtient une contradiction.

Afin de prouver (b), considérons $x \in X_k$ et $y \in E \setminus X_k$ tels que $X_k \cup \{y\} \in \mathcal{F}_1$. Supposons $c_1'(y) > c_1'(x)$. Comme $c_1(y) \leq m_1 \leq c_1(x)$, on doit avoir $x \in R$ et $y \notin R$. Comme $y \in S$, on a $\varepsilon \leq \varepsilon_3 \leq m_1 - c_1(y) \leq c_1(x) - c_1(y) = (c_1'(x) + \varepsilon) - c_1'(y)$ et on obtient une contradiction.

Soit c_2' le poids modifié de c_2, c.-à-d.

$$c_2'(x) := \begin{cases} c_2(x) + \varepsilon & \text{si } x \in R \\ c_2(x) & \text{si } x \notin R \end{cases}.$$

Nous prouvons que X_k et c_2' vérifient les conditions du théorème 13.23 appliqué à $\mathcal{F}_2$.

Afin de prouver (a), considérons $y \in E \setminus X_k$ et $x \in C_2(X_k, y) \setminus \{y\}$. Supposons $c_2'(x) < c_2'(y)$. Comme $c_2(x) \geq c_2(y)$, on doit avoir $y \in R$ et $x \notin R$. Comme $(y, x) \in A^{(2)}$, on a $\varepsilon \leq \varepsilon_2 \leq c_2(x) - c_2(y) = c_2'(x) - (c_2'(y) - \varepsilon)$ et on obtient une contradiction.

Afin de prouver (b), considérons $x \in X_k$ et $y \in E \setminus X_k$ tels que $X_k \cup \{y\} \in \mathcal{F}_2$. Supposons $c_2'(y) > c_2'(x)$. Comme $c_2(y) \leq m_2 \leq c_2(x)$, on doit avoir $y \in R$ et $x \notin R$. Comme $y \in T$, on a $\varepsilon \leq \varepsilon_4 \leq m_2 - c_2(y) \leq c_2(x) - c_2(y) = c_2'(x) - (c_2'(y) - \varepsilon)$ et on obtient une contradiction.

On a ainsi prouvé que (13.5) est bien respecté durant l'exécution de l'étape ⑧ et que l'algorithme répond correctement.

Nous considérons maintenant le temps de calcul. Observons qu'après la mise à jour des poids à l'étape ⑧, les nouveaux ensembles $\bar{S}$, $\bar{T}$, et R, calculés ultérieurement aux étapes ④ et ⑤, sont des sur-ensembles des ensembles $\bar{S}$, $\bar{T}$, et R précédents. Si $\varepsilon = \varepsilon_4 < \infty$, une progression (augmentation de k) s'ensuit. Sinon la cardinalité de R augmente immédiatement d'au moins un (à l'étape ⑤). Ainsi les étapes ④ – ⑧ se répètent moins de $|E|$ fois entre deux augmentations.

Comme le temps de calcul des étapes ④ – ⑧ est $O(|E|^2)$, le temps de calcul total entre deux augmentations est $O(|E|^3)$ plus $O(|E|^2)$ appels à l'oracle (à l'étape ②). Comme il y a $m \leq |E|$ augmentations, le temps de calcul annoncé s'ensuit. □

Le temps de calcul peut être facilement réduit à $O(|E|^3\theta)$ (exercice 22).

Exercices

1. Prouver que les systèmes d'indépendance, exceptés les systèmes (5) et (6), de la liste donnée au début du paragraphe 13.1 ne sont pas – en général – des matroïdes.
2. Montrer que le matroïde uniforme défini sur un ensemble de quatre éléments et de rang 2 n'est pas un matroïde graphique.
3. Prouver qu'un matroïde graphique est représentable sur tout corps.
4. Soit G un graphe non orienté, $K \in \mathbb{N}$ et soit $\mathcal{F}$ la famille des sous-ensembles de $E(G)$ qui sont l'union de K forêts. Prouver que $(E(G), \mathcal{F})$ est un matroïde.
5. Calculer des bornes inférieures serrées pour les rangs quotients des systèmes d'indépendance énumérés au début du paragraphe 13.1.
6. Soit $\mathcal{S}$ une famille d'ensemble. Un ensemble T est un transversal de $\mathcal{S}$ s'il existe une bijection $\Phi : T \to \mathcal{S}$ telle que $t \in \Phi(t)$ pour tout $t \in T$. (Pour une condition nécessaire et suffisante de l'existence d'un transversal, voir exercice 6 du chapitre 10.) Supposons que $\mathcal{S}$ contienne un transversal. Prouver que la famille des transversaux de $\mathcal{S}$ est l'ensemble des bases d'un matroïde.
7. Soit E un ensemble fini et $\mathcal{B} \subseteq 2^E$. Montrer que $\mathcal{B}$ est l'ensemble des bases d'un matroïde $(E, \mathcal{F})$ si et seulement si les conditions suivantes sont vérifiées :

   (B1) $\mathcal{B} \neq \emptyset$;

   (B2) pour tout $B_1, B_2 \in \mathcal{B}$ et $y \in B_2 \setminus B_1$ il existe un $x \in B_1 \setminus B_2$ tel que $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.
8. Soit G un graphe. Soit $\mathcal{F}$ la famille des ensembles $X \subseteq V(G)$, tels qu'il existe un couplage maximum dans G qui ne couvre aucun sommet de X. Prouver que $(V(G), \mathcal{F})$ est un matroïde. Quel est le matroïde dual ?
9. Montrer que $\mathcal{M}(G^*) = (\mathcal{M}(G))^*$ est également vérifié par les graphes G non connexes, ce qui généralise le théorème 13.16.
   Indication : utiliser l'exercice 31(a) du chapitre 2.

10. Montrer que les amas, définis en (3) et (6) dans la liste du paragraphe 13.3, vérifient la propriété flot-max/coupe-min. (Utiliser le théorème 19.10.) Montrer que les amas, définis en (1), (4) et (5), ne vérifient pas la propriété flot-max/coupe-min.

∗ 11. Un amas $(E, \mathcal{F})$ est dit binaire si, pour tout $X_1, \ldots, X_k \in \mathcal{F}$ avec k impair, il existe un $Y \in \mathcal{F}$ tel que $Y \subseteq X_1 \triangle \cdots \triangle X_k$. Prouver que l'amas des T-joints minimaux et l'amas des T-coupes (exemple (7) de la liste du paragraphe 13.3) sont binaires. Prouver qu'un amas est binaire si et seulement si $|A \cap B|$ est impaire pour tout $A \in \mathcal{F}$ et tout $B \in \mathcal{F}^*$, où $(E, \mathcal{F}^*)$ est l'amas bloquant associé. Déduire de cela qu'un amas est binaire si et seulement si son amas bloquant associé est binaire.
Note : Seymour [1977] a classifié les amas binaires avec la propriété flot-max/coupe-min.

∗ 12. Soit P un polyèdre du type bloquant, c.-à-d. tel que $x + y \in P$ pour tout $x \in P$ et $y \geq 0$. Le polyèdre bloquant de P est défini par $B(P) := \{z : z^\top x \geq 1 \text{ pour tout } x \in P\}$. Prouver que $B(P)$ est encore un polyèdre du type bloquant et que $B(B(P)) = P$.
Note : comparer cela avec le théorème 4.22.

13. Comment peut-on vérifier (en temps polynomial) si un ensemble donné d'arêtes d'un graphe complet G est un sous-ensemble d'un cycle hamiltonien de G ?

14. Prouver que si $(E, \mathcal{F})$ est un matroïde, alors l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ maximise toute fonction seuil $c(F) = \min\{c_e : e \in F\}$ sur les bases.

15. Soit $(E, \mathcal{F})$ un matroïde et une fonction $c : E \to \mathbb{R}$ telle que $c(e) \neq c(e')$ pour tout $e \neq e'$ et $c(e) \neq 0$ pour tout e. Prouver que les PROBLÈMES DE MAXIMISATION et de MINIMISATION pour $(E, \mathcal{F}, c)$ ont une solution optimale unique.

∗ 16. Prouver que, pour les matroïdes, les oracles d'indépendance, de sur-ensemble de base, de fermeture et de rang sont polynomialement équivalents.
Indication : pour montrer que l'oracle de rang se réduit à l'oracle d'indépendance, utiliser l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ. Pour montrer que l'oracle d'indépendance se réduit à l'oracle de sur-ensemble de base, utiliser l'algorithme du type GLOUTON-PIRE-SORTI.
(Hausmann et Korte [1981])

17. Étant donné un graphe non orienté G, nous souhaitons colorier les arêtes avec un nombre minimum de couleurs de telle façon que, pour tout cycle C de G, les arêtes de C n'aient pas toutes la même couleur. Montrer qu'il existe un algorithme polynomial pour résoudre ce problème.

18. Soient $(E, \mathcal{F}_1), \ldots, (E, \mathcal{F}_k)$ des matroïdes de fonctions rang $r_1, \ldots, r_k$. Prouver qu'un ensemble $X \subseteq E$ est un ensemble partitionnable si et seulement si $|A| \leq \sum_{i=1}^{k} r_i(A)$ pour tout $A \subseteq X$. Montrer que le théorème 6.19 est un cas particulier.
(Edmonds et Fulkerson [1965])

19. Soit $(E, \mathcal{F})$ un matroïde de fonction rang r. Prouver (en utilisant le théorème 13.34) que :
    (a) $(E, \mathcal{F})$ possède k bases deux à deux disjointes si et seulement si $kr(A) + |E \setminus A| \geq kr(E)$ pour tout $A \subseteq E$.
    (b) $(E, \mathcal{F})$ possède k ensembles indépendants dont l'union est E si et seulement si $kr(A) \geq |A|$ pour tout $A \subseteq E$.

    Montrer que les théorèmes 6.19 et 6.16 sont des cas particuliers.

20. Soient $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$ deux matroïdes. Soit X un sous-ensemble maximal partitionnable relativement à $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2^*) : X = X_1 \dot\cup X_2$ avec $X_1 \in \mathcal{F}_1$ et $X_2 \in \mathcal{F}_2^*$. Soit $B_2 \supseteq X_2$ une base de $\mathcal{F}_2^*$. Prouver qu'alors $X \setminus B_2$ est un ensemble de cardinal maximum dans $\mathcal{F}_1 \cap \mathcal{F}_2$.
    (Edmonds [1970])

21. Soit $(E, \mathcal{S})$ un système d'ensemble et soit $(E, \mathcal{F})$ un matroïde de fonction rang r. Montrer que $\mathcal{S}$ possède un transversal qui est indépendant dans $(E, \mathcal{F})$ si et seulement si $r\left(\bigcup_{B \in \mathcal{B}} B\right) \geq |\mathcal{B}|$ pour tout $\mathcal{B} \subseteq \mathcal{S}$.
    Indication : décrire d'abord la fonction rang du matroïde dont les ensembles indépendants sont tous des transversaux (exercice 6), en utilisant le théorème 13.34. Appliquer ensuite le théorème 13.31.
    (Rado [1942])

22. Montrer que le temps de calcul de l'ALGORITHME DE L'INTERSECTION DE MATROÏDES AVEC POIDS (voir théorème 13.35) peut être réduit à $O(|E|^3\theta)$.

23. Soient $(E, \mathcal{F}_1)$ et $(E, \mathcal{F}_2)$ deux matroïdes et $c : E \to \mathbb{R}$. Soient $X_0, \ldots, X_m \in \mathcal{F}_1 \cap \mathcal{F}_2$ tels que $|X_k| = k$ et $c(X_k) = \max\{c(X) : X \in \mathcal{F}_1 \cap \mathcal{F}_2,\ |X| = k\}$ pour tout k. Prouver que pour $k = 1, \ldots, m-2$

$$c(X_{k+1}) - c(X_k) \ \leq \ c(X_k) - c(X_{k-1}).$$

    (Krogdahl [non publié])

24. Considérons le problème suivant. Étant donné un graphe orienté G avec des poids associés aux arcs, un sommet $s \in V(G)$ et un nombre K, trouver un sous-graphe H de G de poids minimum contenant K chemins arc-disjoints reliant s à tous les autres sommets. Montrer que cela se réduit au PROBLÈME DE L'INTERSECTION DE MATROÏDES AVEC POIDS.
    Indication : voir l'exercice 19 du chapitre 6 et l'exercice 4 de ce chapitre.
    (Edmonds [1970] ; Frank et Tardos [1989] ; Gabow [1995])

25. Soient A et B deux ensembles finis de taille $n \in \mathbb{N}$, $\bar a \in A$, et $c : \{\{a, b\} : a \in A, b \in B\} \to \mathbb{R}$ une fonction coût. Soit $\mathcal{T}$ la famille des ensembles d'arêtes des arbres T tels que $V(T) = A \dot\cup B$ et $|\delta_T(a)| = 2$ pour tout $a \in A \setminus \{\bar a\}$. Montrer qu'un élément de $\mathcal{T}$ de coût minimum peut être calculé en temps $O(n^7)$. Combien d'arêtes seront incidentes à $\bar a$?

Références

Littérature générale :

Bixby, R.E., Cunningham, W.H. [1995] : Matroid optimization and algorithms. In : Handbook of Combinatorics ; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam, 1995

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. [1998] : Combinatorial Optimization. Wiley, New York 1998, Chapter 8

Faigle, U. [1987] : Matroids in combinatorial optimization. In : Combinatorial Geometries (N. White, ed.), Cambridge University Press, 1987

Gondran, M., Minoux, M. [1984] : Graphs and Algorithms. Wiley, Chichester 1984, Chapter 9

Lawler, E.L. [1976] : Combinatorial Optimization ; Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Chapters 7 and 8

Oxley, J.G. [1992] : Matroid Theory. Oxford University Press, Oxford 1992

von Randow, R. [1975] : Introduction to the Theory of Matroids. Springer, Berlin 1975

Recski, A. [1989] : Matroid Theory and its Applications. Springer, Berlin, 1989

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 39–42

Welsh, D.J.A. [1976] : Matroid Theory. Academic Press, London 1976

Références citées :

Cunningham, W.H. [1986] : Improved bounds for matroid partition and intersection algorithms. SIAM Journal on Computing 15 (1986), 948–957

Edmonds, J. [1970] : Submodular functions, matroids and certain polyhedra. In : Combinatorial Structures and Their Applications ; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schonheim, eds.), Gordon and Breach, New York 1970, pp. 69–87

Edmonds, J. [1971] : Matroids and the greedy algorithm. Mathematical Programming 1 (1971), 127–136

Edmonds, J. [1979] : Matroid intersection. In : Discrete Optimization I ; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, eds.), North-Holland, Amsterdam 1979, pp. 39–49

Edmonds, J., Fulkerson, D.R. [1965] : Transversals and matroid partition. Journal of Research of the National Bureau of Standards B 69 (1965), 67–72

Frank, A. [1981] : A weighted matroid intersection algorithm. Journal of Algorithms 2 (1981), 328–336

Frank, A., Tardos, É. [1989] : An application of submodular flows. Linear Algebra and Its Applications 114/115 (1989), 329–348

Fulkerson, D.R. [1971] : Blocking and anti-blocking pairs of polyhedra. Mathematical Programming 1 (1971), 168–194

Gabow, H.N. [1995] : A matroid approach to finding edge connectivity and packing arborescences. Journal of Computer and System Sciences 50 (1995), 259–273

Gabow, H.N., Xu, Y. [1996] : Efficient theoretic and practical algorithms for linear matroid intersection problems. Journal of Computer and System Sciences 53 (1996), 129–147

Hausmann, D., Jenkyns, T.A., Korte, B. [1980] : Worst case analysis of greedy type algorithms for independence systems. Mathematical Programming Study 12 (1980), 120–131

Hausmann, D., Korte, B. [1981] : Algorithmic versus axiomatic definitions of matroids. Mathematical Programming Study 14 (1981), 98–111

Jenkyns, T.A. [1976] : The efficiency of the greedy algorithm. Proceedings of the 7th S-E Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica, Winnipeg 1976, pp. 341–350

Korte, B., Hausmann, D. [1978] : An analysis of the greedy algorithm for independence systems. In : Algorithmic Aspects of Combinatorics ; Annals of Discrete Mathematics 2 (B. Alspach, P. Hell, D.J. Miller, eds.), North-Holland, Amsterdam 1978, pp. 65–74

Korte, B., Monma, C.L. [1979] : Some remarks on a classification of oracle-type algorithms. In : Numerische Methoden bei graphentheoretischen und kombinatorischen Problemen ; Band 2 (L. Collatz, G. Meinardus, W. Wetterling, eds.), Birkhäuser, Basel 1979, pp. 195–215

Lehman, A. [1979] : On the width-length inequality. Mathematical Programming 17 (1979), 403–417

Nash-Williams, C.S.J.A. [1967] : An application of matroids to graph theory. In : Theory of Graphs ; Proceedings of an International Symposium in Rome 1966 (P. Rosenstiehl, ed.), Gordon and Breach, New York, 1967, pp. 263–265

Rado, R. [1942] : A theorem on independence relations. Quarterly Journal of Math. Oxford 13 (1942), 83–89

Rado, R. [1957] : Note on independence functions. Proceedings of the London Mathematical Society 7 (1957), 300–320

Seymour, P.D. [1977] : The matroids with the Max-Flow Min-Cut property. Journal of Combinatorial Theory B 23 (1977), 189–222

Whitney, H. [1933] : Planar graphs. Fundamenta Mathematicae 21 (1933), 73–84

Whitney, H. [1935] : On the abstract properties of linear dependence. American Journal of Mathematics 57 (1935), 509–533

Chapitre 14

Généralisations des matroïdes

Il existe plusieurs généralisations intéressantes des matroïdes. Nous avons déjà vu les systèmes d'indépendance au paragraphe 13.1, obtenus lorsque l'on omet l'axiome (M3). Au paragraphe 14.1, nous considérons les greedoïdes, obtenus en supprimant (M2) à la place de (M3). De plus, l'étude de certains polytopes liés aux matroïdes et aux fonctions sous-modulaires, appelés polymatroïdes, mène à de puissantes généralisations de théorèmes importants ; nous décrirons ces polytopes au paragraphe 14.2. Aux paragraphes 14.3 et 14.4, nous considérons deux approches du problème de la minimisation d'une fonction sous-modulaire arbitraire : l'une utilisant la MÉTHODE DES ELLIPSOÏDES, et l'autre un algorithme combinatoire. Pour le cas particulier important des fonctions sous-modulaires symétriques, nous mentionnons un algorithme plus simple au paragraphe 14.5.

14.1 Greedoïdes

Par définition, les systèmes d'ensembles $(E, \mathcal{F})$ sont des matroïdes si et seulement s'ils vérifient :

(M1) $\emptyset \in \mathcal{F}$;

(M2) si $X \subseteq Y \in \mathcal{F}$ alors $X \in \mathcal{F}$;

(M3) si $X, Y \in \mathcal{F}$ et $|X| > |Y|$, alors il existe un $x \in X \setminus Y$ tel que $Y \cup \{x\} \in \mathcal{F}$.

Si nous supprimons (M3), nous obtenons des systèmes d'indépendance, décrits aux paragraphes 13.1 et 13.4. Nous supprimons maintenant (M2) :

Définition 14.1. *Un* **greedoïde** *est un système d'ensembles* $(E, \mathcal{F})$ *vérifiant* (M1) *et* (M3).

Au lieu de la propriété de sous-inclusion (M2), nous avons une propriété d'accessibilité : on dit qu'un système d'ensembles $(E, \mathcal{F})$ est **accessible** si $\emptyset \in \mathcal{F}$ et pour tout $X \in \mathcal{F} \setminus \{\emptyset\}$, il existe un $x \in X$ tel que $X \setminus \{x\} \in \mathcal{F}$. Les greedoïdes sont

accessibles (l'accessibilité est une conséquence directe de (M1) et (M3)). Bien que plus généraux que les matroïdes, ils possèdent une structure riche et, par ailleurs, ils généralisent de nombreux concepts différents, en apparence indépendants. Nous commençons par le résultat suivant :

Théorème 14.2. *Soit $(E, \mathcal{F})$ un système d'ensembles accessible. Les affirmations suivantes sont équivalentes :*

(a) *Pour tout $X \subseteq Y \subset E$ et $z \in E \setminus Y$ tel que $X \cup \{z\} \in \mathcal{F}$ et $Y \in \mathcal{F}$, on a $Y \cup \{z\} \in \mathcal{F}$.*

(b) *$\mathcal{F}$ est fermé pour l'union.*

Preuve. (a) $\Rightarrow$(b) : Soient $X, Y \in \mathcal{F}$; nous montrons que $X \cup Y \in \mathcal{F}$. Soit Z un ensemble de taille maximale tel que $Z \in \mathcal{F}$ et $X \subseteq Z \subseteq X \cup Y$. Supposons $Y \setminus Z \neq \emptyset$. Par des applications successives de la propriété d'accessibilité à Y, nous obtenons un ensemble $Y' \in \mathcal{F}$ tel que $Y' \subseteq Z$ et un élément $y \in Y \setminus Z$ tel que $Y' \cup \{y\} \in \mathcal{F}$. En appliquant alors (a) à Z, Y' et y, nous obtenons $Z \cup \{y\} \in \mathcal{F}$, ce qui contredit le choix de Z.

(b) $\Rightarrow$(a) est évident. □

Si les conditions du théorème 14.2 sont vérifiées, alors $(E, \mathcal{F})$ est appelé un **antimatroïde**.

Proposition 14.3. *Tout antimatroïde est un greedoïde.*

Preuve. Soit $(E, \mathcal{F})$ un antimatroïde, c.-à-d. accessible et fermé pour l'union. Afin de prouver (M3), considérons $X, Y \in \mathcal{F}$ tels que $|X| > |Y|$. Comme $(E, \mathcal{F})$ est accessible, il existe un ordre $X = \{x_1, \ldots, x_n\}$ tel que $\{x_1, \ldots, x_i\} \in \mathcal{F}$ pour $i = 0, \ldots, n$. Soit $i \in \{1, \ldots, n\}$ l'indice minimum tel que $x_i \notin Y$; alors $Y \cup \{x_i\} = Y \cup \{x_1, \ldots, x_i\} \in \mathcal{F}$ (puisque $\mathcal{F}$ est fermé pour l'union). □

On peut donner une autre définition des antimatroïdes, équivalente à la précédente, à l'aide d'un opérateur de fermeture :

Proposition 14.4. *Soit $(E, \mathcal{F})$ un système d'ensembles tel que $\mathcal{F}$ est fermé pour l'union et $\emptyset \in \mathcal{F}$. Définissons*

$$\tau(A) := \bigcap \{X \subseteq E : A \subseteq X,\ E \setminus X \in \mathcal{F}\}.$$

Alors τ est un opérateur de fermeture, c.-à-d. vérifie (S1)–(S3) du théorème 13.11.

Preuve. Soit $X \subseteq Y \subseteq E$. $X \subseteq \tau(X) \subseteq \tau(Y)$ est évident. Afin de prouver (S3), supposons qu'il existe un $y \in \tau(\tau(X)) \setminus \tau(X)$. Alors $y \in Y$ pour tout $Y \subseteq E$ tel que $\tau(X) \subseteq Y$ et $E \setminus Y \in \mathcal{F}$, mais il existe un $Z \subseteq E \setminus \{y\}$ tel que $X \subseteq Z$ et $E \setminus Z \in \mathcal{F}$. Cela implique $\tau(X) \not\subseteq Z$, une contradiction. □

Théorème 14.5. *Soit $(E, \mathcal{F})$ un système d'ensembles tel que $\mathcal{F}$ est fermé pour l'union et $\emptyset \in \mathcal{F}$. Alors $(E, \mathcal{F})$ est accessible si et seulement si l'opérateur de fermeture τ de la proposition 14.4 vérifie la propriété d'antiéchange : si $X \subseteq E$, $y, z \in E \setminus \tau(X)$, $y \neq z$ et $z \in \tau(X \cup \{y\})$, alors $y \notin \tau(X \cup \{z\})$.*

Preuve. Si $(E, \mathcal{F})$ est accessible, alors (M3) est vérifié d'après la proposition 14.3. Afin de montrer la propriété d'antiéchange, considérons $X \subseteq E$, $B := E \setminus \tau(X)$, et $y, z \in B$ tels que $z \notin A := E \setminus \tau(X \cup \{y\})$. Remarquons que $A \in \mathcal{F}$, $B \in \mathcal{F}$ et $A \subseteq B \setminus \{y, z\}$.

En appliquant (M3) à A et B, nous obtenons un élément $b \in B \setminus A \subseteq E \setminus (X \cup A)$ tel que $A \cup \{b\} \in \mathcal{F}$. $A \cup \{b\}$ ne peut pas être un sous-ensemble de $E \setminus (X \cup \{y\})$ (sinon $\tau(X \cup \{y\}) \subseteq E \setminus (A \cup \{b\})$, ce qui contredit $\tau(X \cup \{y\}) = E \setminus A$). Ainsi $b = y$. On a donc $A \cup \{y\} \in \mathcal{F}$ et ainsi $\tau(X \cup \{z\}) \subseteq E \setminus (A \cup \{y\})$. Nous avons prouvé que $y \notin \tau(X \cup \{z\})$.

Pour montrer le sens contraire, considérons $A \in \mathcal{F} \setminus \{\emptyset\}$ et $X := E \setminus A$. On a $\tau(X) = X$. Soit $a \in A$ tel que $|\tau(X \cup \{a\})|$ soit minimum. Nous affirmons que $\tau(X \cup \{a\}) = X \cup \{a\}$, c.-à-d. $A \setminus \{a\} \in \mathcal{F}$.

Supposons qu'au contraire $b \in \tau(X \cup \{a\}) \setminus (X \cup \{a\})$. D'après (c), on a $a \notin \tau(X \cup \{b\})$. De plus,

$$\tau(X \cup \{b\}) \subseteq \tau(\tau(X \cup \{a\}) \cup \{b\}) = \tau(\tau(X \cup \{a\})) = \tau(X \cup \{a\}).$$

Ainsi $\tau(X \cup \{b\})$ est un sous-ensemble propre de $\tau(X \cup \{a\})$, ce qui contredit le choix de a. □

La propriété d'antiéchange du théorème 14.5 est différente de (S4). Alors que la propriété (S4), du théorème 13.11, concerne les enveloppes linéaires dans $\mathbb{R}^n$, la propriété d'antiéchange concerne les enveloppes convexes dans $\mathbb{R}^n$: si $y \neq z$, $z \notin \text{conv}(X)$ et $z \in \text{conv}(X \cup \{y\})$, alors évidemment $y \notin \text{conv}(X \cup \{z\})$. Donc pour tout ensemble fini $E \subset \mathbb{R}^n$, $(E, \{X \subseteq E : X \cap \text{conv}(E \setminus X) = \emptyset\})$ est un antimatroïde.

Les greedoïdes généralisent les matroïdes et les antimatroïdes, mais ils incluent aussi d'autres structures intéressantes. Un premier exemple est la structure de blossoms que nous utilisons dans l'ALGORITHME DE COUPLAGE MAXIMUM D'EDMONDS (exercice 1). Voici un autre exemple de base :

Proposition 14.6. *Soient G un graphe (orienté ou non orienté) et $r \in V(G)$. Soit $\mathcal{F}$ la famille des ensembles d'arcs des arborescences de G de racine r, ou des ensembles des arêtes des arbres de G contenant r (non nécessairement couvrants). Alors $(E(G), \mathcal{F})$ est un greedoïde.*

Preuve. (M1) est évident. Nous prouvons (M3) dans le cas orienté ; le cas non orienté se prouve de la même façon. Soient (X_1, F_1) et (X_2, F_2) deux arborescences de G de racine r telles que $|F_1| > |F_2|$. Alors $|X_1| = |F_1| + 1 > |F_2| + 1 = |X_2|$. Considérons donc $x \in X_1 \setminus X_2$. Le chemin de r à x dans (X_1, F_1) contient un arc (v, w) tel que $v \in X_2$ et $w \notin X_2$. Cet arc peut être ajouté à (X_2, F_2), ce qui prouve que $F_2 \cup \{(v, w)\} \in \mathcal{F}$. □

Ce greedoïde est appelé le greedoïde des ramifications orientées (non orientées) de G.

Le problème de la recherche d'un arbre couvrant de poids maximum dans un graphe connexe G, avec des poids non négatifs, correspond au PROBLÈME

DE MAXIMISATION appliqué au matroïde des cycles $\mathcal{M}(G)$. Dans ce cas, nous voyons que L'ALGORITHME GLOUTON-MEILLEUR-INSÉRÉ n'est rien d'autre que l'ALGORITHME DE KRUSKAL. Nous avons maintenant une seconde formulation du même problème : nous recherchons un ensemble de poids maximum F tel que $F \in \mathcal{F}$, où $(E(G), \mathcal{F})$ est le greedoïde des ramifications non orientées de G.

Nous formulons maintenant un algorithme glouton général pour les greedoïdes. Dans le cas particulier des matroïdes, il correspond exactement à l'ALGORITHME GLOUTON-MEILLEUR-INSÉRÉ décrit au paragraphe 13.4. Si nous considérons un greedoïde de ramifications non orientées et une fonction coût modulaire c, il correspond à l'ALGORITHME DE PRIM :

ALGORITHME GLOUTON POUR LES GREEDOÏDES

Input Un greedoïde $(E, \mathcal{F})$ et une fonction $c : 2^E \to \mathbb{R}$, donnés par un oracle qui, pour tout sous-ensemble $X \subseteq E$, dit si $X \in \mathcal{F}$ et retourne $c(X)$.

Output Un ensemble $F \in \mathcal{F}$.

① Poser $F := \emptyset$.

② Soit $e \in E \setminus F$ tel que $F \cup \{e\} \in \mathcal{F}$ et $c(F \cup \{e\})$ soit maximum ;
if il n'existe pas un tel e **then stop**.

③ Poser $F := F \cup \{e\}$ et **go to** ②.

Cet algorithme ne retourne pas toujours une solution optimale, même pour des fonctions coûts c modulaires. Nous pouvons au moins caractériser les greedoïdes pour lesquels l'algorithme répond correctement :

Théorème 14.7. *Soit $(E, \mathcal{F})$ un greedoïde. L'*ALGORITHME GLOUTON POUR LES GREEDOÏDES *trouve un ensemble $F \in \mathcal{F}$ de poids maximum pour toute fonction poids modulaire $c : 2^E \to \mathbb{R}_+$ si et seulement si $(E, \mathcal{F})$ vérifie la propriété dite d'échange fort : pour tout $A \in \mathcal{F}$, B maximal dans $\mathcal{F}$, $A \subseteq B$ et $x \in E \setminus B$ tels que $A \cup \{x\} \in \mathcal{F}$, il existe un $y \in B \setminus A$ tel que $A \cup \{y\} \in \mathcal{F}$ et $(B \setminus y) \cup \{x\} \in \mathcal{F}$.*

Preuve. Supposons que $(E, \mathcal{F})$ soit un greedoïde vérifiant la propriété d'échange fort. Soit $c : E \to \mathbb{R}_+$, et soit $A = \{a_1, \ldots, a_l\}$ la solution trouvée par l'ALGORITHME GLOUTON POUR LES GREEDOÏDES, où les éléments sont pris dans l'ordre $a_1, \ldots, a_l$.

Soit $B = \{a_1, \ldots, a_k\} \dot\cup B'$ une solution optimale telle que k soit maximum, et supposons que $k < l$. Nous appliquons alors la propriété d'échange fort à $\{a_1, \ldots, a_k\}$, B et a_{k+1}. Nous en déduisons qu'il existe un $y \in B'$ tel que $\{a_1, \ldots, a_k, y\} \in \mathcal{F}$ et $(B \setminus y) \cup \{a_{k+1}\} \in \mathcal{F}$. D'après le choix de a_{k+1} à l'étape ② de l'ALGORITHME GLOUTON POUR LES GREEDOÏDES, on a $c(a_{k+1}) \geq c(y)$ et ainsi $c((B \setminus y) \cup \{a_{k+1}\}) \geq c(B)$, ce qui contredit le choix de B.

Pour le sens inverse, soit $(E, \mathcal{F})$ un greedoïde ne vérifiant pas la propriété d'échange fort. Soit $A \in \mathcal{F}$, B maximal dans $\mathcal{F}$, $A \subseteq B$ et $x \in E \setminus B$ tel que $A \cup \{x\} \in \mathcal{F}$ et pour tout $y \in B \setminus A$, tel que $A \cup \{y\} \in \mathcal{F}$, on ait $(B \setminus y) \cup \{x\} \notin \mathcal{F}$.

Soit $Y := \{y \in B \setminus A : A \cup \{y\} \in \mathcal{F}\}$. On pose $c(e) := 2$ pour $e \in B \setminus Y$, et $c(e) := 1$ pour $e \in Y \cup \{x\}$ et $c(e) := 0$ pour $e \in E \setminus (B \cup \{x\})$. Alors l'ALGORITHME GLOUTON POUR LES GREEDOÏDES devrait choisir d'abord les éléments de A (ils ont des poids égaux à 2) et ensuite, il devrait choisir x. Il finira finalement avec un ensemble $F \in \mathcal{F}$ qui peut ne pas être optimal, puisque $c(F) \leq c(B \cup \{x\}) - 2 < c(B \cup \{x\}) - 1 = c(B)$ et $B \in \mathcal{F}$. □

En effet, l'optimisation de fonctions modulaires sur des greedoïdes généraux est *NP*-difficile. Cela est une conséquence de la proposition suivante (et du corollaire 15.24) :

Proposition 14.8. *Le problème de décider, pour un graphe non orienté G donné et $k \in \mathbb{N}$, si G possède une couverture par les sommets, de cardinal k, se réduit linéairement au problème suivant : étant donné un greedoïde $(E, \mathcal{F})$ (par un oracle d'appartenance) et une fonction $c : E \to \mathbb{R}_+$, trouver un ensemble $F \in \mathcal{F}$ tel que $c(F)$ soit maximum.*

Preuve. Soit G un graphe non orienté et $k \in \mathbb{N}$. Soient $D := V(G) \dot\cup E(G)$ et

$$\mathcal{F} := \{X \subseteq D : \text{ pour tout } e = (v, w) \in E(G) \cap X, \text{ on a } v \in X \text{ ou } w \in X\}.$$

$(D, \mathcal{F})$ est un antimatroïde : il est accessible et fermé pour l'union. En particulier, d'après la proposition 14.3, c'est un greedoïde.

Considérons maintenant $\mathcal{F}' := \{X \in \mathcal{F} : |X| \leq |E(G)| + k\}$. Comme (M1) et (M3) sont préservés, $(D, \mathcal{F}')$ est également un greedoïde. Posons $c(e) := 1$ pour $e \in E(G)$ et $c(v) := 0$ pour $v \in V(G)$. Alors il existe un ensemble $F \in \mathcal{F}'$ tel que $c(F) = |E(G)|$ si et seulement si G contient une couverture par les sommets de taille k. □

D'autre part, il existe des fonctions intéressantes qui peuvent être maximisées sur des greedoïdes arbitraires, par exemple des fonctions seuils $c(F) := \min\{c'(e) : e \in F\}$ pour $c' : E \to \mathbb{R}_+$ (exercice 2). Voir Korte, Lovász et Schrader [1991] pour davantage de résultats sur ce sujet.

14.2 Polymatroïdes

Depuis le théorème 13.10, nous connaissons le lien étroit entre les matroïdes et les fonctions sous-modulaires. Les fonctions sous-modulaires permettent de définir la classe intéressante de polyèdres suivante :

Définition 14.9. *Un* **polymatroïde** *est un polytope du type*

$$P(f) := \left\{ x \in \mathbb{R}^E : x \geq 0, \sum_{e \in A} x_e \leq f(A) \text{ pour tout } A \subseteq E \right\}$$

où E est un ensemble fini et $f : 2^E \to \mathbb{R}_+$ est une fonction sous-modulaire.

Il n'est pas difficile de voir que tout polymatroïde f peut être choisi de telle sorte que $f(\emptyset) = 0$ et que f soit monotone (exercice 5 ; une fonction $f : 2^E \to \mathbb{R}$ est dite **monotone** si $f(X) \leq f(Y)$ pour $X \subseteq Y \subseteq E$). La définition initiale d'Edmonds était différente ; voir l'exercice 6. De plus, mentionnons que le terme polymatroïde n'est parfois pas utilisé pour désigner le polytope mais le couple (E, f).

Si f est la fonction rang d'un matroïde, $P(f)$ est l'enveloppe convexe des vecteurs d'incidence des ensembles indépendants de ce matroïde (théorème 13.21). Nous savons que l'algorithme du type GLOUTON-MEILLEUR-INSÉRÉ permet d'optimiser toute fonction linéaire sur le polytope des matroïdes. Un algorithme glouton similaire fonctionne également pour les polymatroïdes. Nous supposons que f est monotone :

ALGORITHME GLOUTON POUR LES POLYMATROÏDES

Input Un ensemble fini E et une fonction sous-modulaire et monotone $f : 2^E \to \mathbb{R}_+$ (donnée par un oracle). Un vecteur $c \in \mathbb{R}^E$.

Output Un vecteur $x \in P(f)$ tel que cx soit maximum.

① Trier $E = \{e_1, \ldots, e_n\}$ de telle façon que $c(e_1) \geq \cdots \geq c(e_k) > 0 \geq c(e_{k+1}) \geq \cdots \geq c(e_n)$.

② **If** $k \geq 1$ **then** poser $x(e_1) := f(\{e_1\})$.
Poser $x(e_i) := f(\{e_1, \ldots, e_i\}) - f(\{e_1, \ldots, e_{i-1}\})$ pour $i = 2, \ldots, k$.
Poser $x(e_i) := 0$ pour $i = k+1, \ldots, n$.

Proposition 14.10. *Soient $E = \{e_1, \ldots, e_n\}$ et $f : 2^E \to \mathbb{R}$ une fonction sous-modulaire telle que $f(\emptyset) \geq 0$. Soit $b : E \to \mathbb{R}$ tel que $b(e_1) \leq f(\{e_1\})$ et $b(e_i) \leq f(\{e_1, \ldots, e_i\}) - f(\{e_1, \ldots, e_{i-1}\})$ pour $i = 2, \ldots, n$. Alors $\sum_{a \in A} b(a) \leq f(A)$ pour tout $A \subseteq E$.*

Preuve. Par induction sur $i = \max\{j : e_j \in A\}$. L'affirmation est évidente pour $A = \emptyset$ et $A = \{e_1\}$. Si $i \geq 2$, alors $\sum_{a \in A} b(a) = \sum_{a \in A \setminus \{e_i\}} b(a) + b(e_i) \leq f(A \setminus \{e_i\}) + b(e_i) \leq f(A \setminus \{e_i\}) + f(\{e_1, \ldots, e_i\}) - f(\{e_1, \ldots, e_{i-1}\}) \leq f(A)$, où la première inégalité provient de l'hypothèse d'induction et la troisième est conséquence de la sous-modularité. □

Théorème 14.11. *L'*ALGORITHME GLOUTON POUR LES POLYMATROÏDES *répond correctement en trouvant un $x \in P(f)$ tel que cx soit maximum. Si f est entier, alors x est également entier.*

Preuve. Soit $x \in \mathbb{R}^E$ l'output de l'ALGORITHME GLOUTON POUR LES POLYMATROÏDES pour E, f et c. Par définition, si f est entier, alors x est également entier. On a $x \geq 0$ puisque f est monotone et ainsi $x \in P(f)$, d'après la proposition 14.10.

Considérons maintenant $y \in \mathbb{R}_+^E$ tel que $cy > cx$. Comme pour la preuve du théorème 13.19, nous posons $d_j := c(e_j) - c(e_{j+1})$ $(j = 1, \ldots, k-1)$ et $d_k := c(e_k)$, et obtenons

$$\sum_{j=1}^{k} d_j \sum_{i=1}^{j} x(e_i) \;=\; cx \;<\; cy \;\leq\; \sum_{j=1}^{k} c(e_j)y(e_j) \;=\; \sum_{j=1}^{k} d_j \sum_{i=1}^{j} y(e_i).$$

Comme $d_j \geq 0$ pour tout j, il existe un indice $j \in \{1, \ldots, k\}$ tel que $\sum_{i=1}^{j} y(e_i) > \sum_{i=1}^{j} x(e_i)$; cependant, comme $\sum_{i=1}^{j} x(e_i) = f(\{e_1, \ldots, e_j\})$ cela signifie que $y \notin P(f)$. □

Comme dans le cas des matroïdes, on peut aussi traiter le problème de l'intersection de deux polymatroïdes. Le théorème de l'intersection de polymatroïdes suivant a de nombreuses implications :

Théorème 14.12. (Edmonds [1970,1979]) *Soit E un ensemble fini et soient $f, g : 2^E \to \mathbb{R}_+$ des fonctions sous-modulaires. Alors le système*

$$\begin{aligned} x &\geq 0 \\ \sum_{e\in A} x_e &\leq f(A) \qquad (A \subseteq E) \\ \sum_{e\in A} x_e &\leq g(A) \qquad (A \subseteq E) \end{aligned}$$

est TDI.

Preuve. Considérons les deux PLs duaux suivants :

$$\max \left\{ cx : \sum_{e\in A} x_e \leq f(A) \text{ et } \sum_{e\in A} x_e \leq g(A) \text{ pour tout } A \subseteq E,\ x \geq 0 \right\}$$

et

$$\min \left\{ \sum_{A\subseteq E} (f(A)y_A + g(A)z_A) : \sum_{A\subseteq E,\, e\in A} (y_A + z_A) \geq c_e \text{ pour tout } e \in E,\ y, z \geq 0 \right\}.$$

Afin de montrer la totale dual-intégralité, nous utilisons le lemme 5.23.

Soit $c : E(G) \to \mathbb{Z}$, et soit y, z une solution duale optimale pour laquelle

$$\sum_{A\subseteq E} (y_A + z_A)|A||E \setminus A| \tag{14.1}$$

est aussi petit que possible. Nous affirmons que $\mathcal{F} := \{A \subseteq E : y_A > 0\}$ est une chaîne, c.-à-d. pour tout $A, B \in \mathcal{F}$ soit $A \subseteq B$, soit $B \subseteq A$.

Pour voir cela, supposons que $A, B \in \mathcal{F}$ sont tels que $A \cap B \neq A$ et $A \cap B \neq B$. Soit $\epsilon := \min\{y_A, y_B\}$. Posons $y'_A := y_A - \epsilon$, $y'_B := y_B - \epsilon$, $y'_{A\cap B} := y_{A\cap B} + \epsilon$, $y'_{A\cup B} := y_{A\cup B} + \epsilon$, et $y'(S) := y(S)$ pour tout autre $S \subseteq E$. Comme y', z est une solution duale réalisable, elle est également optimale (f est sous-modulaire) et contredit le choix de y, car (14.1) est plus petit pour y', z.

Par le même argument, $\mathcal{F}' := \{A \subseteq E : z_A > 0\}$ est une chaîne. Soient maintenant M et M' les matrices dont les colonnes sont indicées par les éléments de E et les lignes par les vecteurs d'incidence des éléments de, respectivement, $\mathcal{F}$ et $\mathcal{F}'$. D'après le lemme 5.23, il suffit de montrer que $\binom{M}{M'}$ est totalement unimodulaire.

Ici nous utilisons le théorème de Ghouila-Houri 5.24. Soit $\mathcal{R}$ un ensemble de lignes, disons $\mathcal{R} = \{A_1, \ldots, A_p, B_1, \ldots, B_q\}$ tel que $A_1 \supseteq \cdots \supseteq A_p$ et $B_1 \supseteq \cdots \supseteq B_q$. Soit $\mathcal{R}_1 := \{A_i : i \text{ impair}\} \cup \{B_i : i \text{ pair}\}$ et $\mathcal{R}_2 := \mathcal{R} \setminus \mathcal{R}_1$. Comme pour tout $e \in E$, on a $\{R \in \mathcal{R} : e \in R\} = \{A_1, \ldots, A_{p_e}\} \cup \{B_1, \ldots, B_{q_e}\}$ pour un certain $p_e \in \{0, \ldots, p\}$ et un certain $q_e \in \{0, \ldots, q\}$, la somme des lignes de $\mathcal{R}_1$ moins la somme des lignes de $\mathcal{R}_2$ est un vecteur dont les composantes sont seulement $-1, 0, 1$. Donc le critère du théorème 5.24 est vérifié. □

On peut optimiser des fonctions linéaires sur l'intersection de deux polymatroïdes. Cependant, cela n'est pas aussi facile qu'avec un seul polymatroïde. Mais on peut utiliser la MÉTHODE DES ELLIPSOÏDES si on peut résoudre le PROBLÈME DE SÉPARATION pour tout polymatroïde. Nous reviendrons à cette question au paragraphe 14.3.

Corollaire 14.13. (Edmonds [1970]) *Soient $(E, \mathcal{M}_1)$ et $(E, \mathcal{M}_2)$ deux matroïdes de fonctions rang r_1 et r_2. Alors l'enveloppe convexe des vecteurs d'incidence des éléments de $\mathcal{M}_1 \cap \mathcal{M}_2$ est le polytope*

$$\left\{ x \in \mathbb{R}_+^E : \sum_{e \in A} x_e \leq \min\{r_1(A), r_2(A)\} \text{ pour tout } A \subseteq E \right\}.$$

Preuve. Comme r_1 et r_2 sont des fonctions non négatives et sous-modulaires (d'après le théorème 13.10), le système d'inégalités précédent est TDI (d'après le théorème 14.12). Comme r_1 et r_2 sont à valeurs entières, le polytope est entier (d'après le corollaire 5.15). Comme $r_1(A) \leq |A|$ pour tout $A \subseteq E$, les sommets (le polytope en est l'enveloppe convexe d'après le corollaire 3.32) sont des vecteurs 0-1, de même que les vecteurs d'incidence des ensembles indépendants communs (éléments de $\mathcal{M}_1 \cap \mathcal{M}_2$). D'autre part, chacun de ces vecteurs d'incidence vérifie les inégalités (par définition de la fonction rang). □

Bien entendu, la description du polytope des matroïdes (théorème 13.21) découle de cela en posant $\mathcal{M}_1 = \mathcal{M}_2$. Le théorème 14.12 a d'autres conséquences :

Corollaire 14.14. (Edmonds [1970]) *Soit E un ensemble fini et soient $f, g : 2^E \to \mathbb{R}_+$ des fonctions sous-modulaires et monotones. Alors*

$$\max\{\mathbb{1}x : x \in P(f) \cap P(g)\} \;=\; \min_{A \subseteq E}(f(A) + g(E \setminus A)).$$

De plus, si f et g sont à valeurs entières, alors il existe un x entier pour lequel le maximum est atteint.

Preuve. D'après le théorème 14.12, le dual de

$$\max\{\mathbb{1}x : x \in P(f) \cap P(g)\},$$

qui est

$$\min\left\{\sum_{A\subseteq E}(f(A)y_A + g(A)z_A) : \sum_{A\subseteq E,\, e\in A}(y_A + z_A) \geq 1 \text{ pour tout } e \in E,\ y, z \geq 0\right\}$$

possède une solution optimale entière y, z. Soit $B := \bigcup_{A:y_A\geq 1} A$ et $C := \bigcup_{A:z_A\geq 1} A$. Posons $y'_B := 1$, $z'_C := 1$ et mettons toutes les autres composantes de y' et z' à zéro. On a $B \cup C = E$ et y', z' est une solution duale réalisable. Comme f et g sont sous-modulaires et non négatives,

$$\sum_{A\subseteq E}(f(A)y_A + g(A)z_A) \ \geq\ f(B) + g(C).$$

Comme $E \setminus B \subseteq C$ et comme g est monotone, cela est supérieur ou égal à $f(B) + g(E \setminus B)$, ce qui prouve le sens $\geq$.

L'autre inégalité $\leq$ est évidente, car pour tout $A \subseteq E$, on obtient une solution duale réalisable y, z en posant $y_A := 1$, $z_{E\setminus A} := 1$ et en mettant toutes les autres composantes à zéro.

L'intégralité est une conséquence directe du théorème 14.12 et du corollaire 5.15. □

Le théorème 13.31 est un cas particulier du précédent. De plus, on obtient :

Corollaire 14.15. (Frank [1982]) *Soient E un ensemble fini et $f, g : 2^E \to \mathbb{R}$ deux fonctions telles que f soit supermodulaire, g soit sous-modulaire et $f \leq g$. Alors il existe une fonction modulaire $h : 2^E \to \mathbb{R}$ telle que $f \leq h \leq g$. Si f et g sont à valeurs entières, h peut être choisi à valeurs entières.*

Preuve. Soit $M := 2\max\{|f(A)| + |g(A)| : A \subseteq E\}$. Posons $f'(A) := g(E) - f(E \setminus A) + M|A|$ et $g'(A) := g(A) - f(\emptyset) + M|A|$ pour tout $A \subseteq E$. f' et g' sont non négatives, sous-modulaires et monotones. L'application du corollaire 14.14 donne

$$\begin{aligned}
& \max\{\mathbb{1}x : x \in P(f') \cap P(g')\} \\
= \ & \min_{A\subseteq E}(f'(A) + g'(E \setminus A)) \\
= \ & \min_{A\subseteq E}(g(E) - f(E \setminus A) + M|A| + g(E \setminus A) - f(\emptyset) + M|E \setminus A|) \\
\geq \ & g(E) - f(\emptyset) + M|E|.
\end{aligned}$$

Considérons donc $x \in P(f') \cap P(g')$ tel que $\mathbb{1}x = g(E) - f(\emptyset) + M|E|$. Si f et g sont à valeurs entières, x peut être choisi entier. Posons $h'(A) := \sum_{e\in A} x_e$ et $h(A) := h'(A) + f(\emptyset) - M|A|$ pour tout $A \subseteq E$. h est modulaire. De plus, pour tout $A \subseteq E$, on a $h(A) \leq g'(A) + f(\emptyset) - M|A| = g(A)$ et $h(A) = \mathbb{1}x - h'(E \setminus A) + f(\emptyset) - M|A| \geq g(E) + M|E| - M|A| - f'(E \setminus A) = f(A)$. □

L'analogie avec les fonctions convexes et concaves est évidente ; voir également l'exercice 9.

14.3 Minimisation de fonctions sous-modulaires

Le PROBLÈME DE SÉPARATION pour un polymatroïde $P(f)$ et un vecteur x correspond à la recherche d'un ensemble A tel que $f(A) < \sum_{e\in A} x(e)$. Donc ce problème se réduit à la recherche d'un ensemble A minimisant $g(A)$, où $g(A) := f(A) - \sum_{e\in A} x(e)$. Remarquons que si f est sous-modulaire, alors g est également sous-modulaire. Ainsi le problème de la minimisation d'une fonction sous-modulaire est un problème intéressant.

Une autre motivation pourrait être que les fonctions sous-modulaires peuvent être considérées comme l'analogue discret des fonctions convexes (corollaire 14.15 et exercice 9). Nous avons déjà résolu un cas particulier au paragraphe 8.7 : trouver une coupe minimum dans un graphe non orienté correspond à minimiser une certaine fonction sous-modulaire symétrique $f : 2^U \to \mathbb{R}_+$, sur $2^U \setminus \{\emptyset, U\}$. Avant de revenir à ce cas particulier, nous montrons d'abord comment minimiser des fonctions sous-modulaires générales. Nous supposons qu'une borne supérieure de $\text{taille}(f(S))$ est donnée. Pour simplifier, nous nous restreignons aux fonctions sous-modulaires à valeurs entières :

PROBLÈME DE MINIMISATION D'UNE FONCTION SOUS-MODULAIRE	
Instance	Un ensemble fini U. Une fonction sous-modulaire $f : 2^U \to \mathbb{Z}$ (donnée par un oracle).
Tâche	Trouver un sous-ensemble $X \subseteq U$ tel que $f(X)$ soit minimum.

Grötschel, Lovász et Schrijver [1981] ont montré comment ce problème peut être résolu à l'aide de la MÉTHODE DES ELLIPSOÏDES. L'idée est de déterminer le minimum par recherche dichotomique ; cela réduit le problème au PROBLÈME DE SÉPARATION pour un polymatroïde. En utilisant l'équivalence entre séparation et optimisation (paragraphe 4.6), il suffit ainsi d'optimiser des fonctions linéaires sur les polymatroïdes. Cela peut se faire facilement à l'aide de l'ALGORITHME GLOUTON POUR LES POLYMATROÏDES. Nous avons tout d'abord besoin d'une borne supérieure de $|f(S)|$ pour $S \subseteq U$:

Proposition 14.16. *Pour toute fonction sous-modulaire $f : 2^U \to \mathbb{Z}$ et tout $S \subseteq U$, on a*

$$f(U) - \sum_{u\in U} \max\{0, f(\{u\}) - f(\emptyset)\} \;\le\; f(S) \;\le\; f(\emptyset) + \sum_{u\in U} \max\{0, f(\{u\}) - f(\emptyset)\}.$$

En particulier, un nombre B tel que $|f(S)| \le B$ pour tout $S \subseteq U$, peut être calculé en temps linéaire, à l'aide de $|U| + 2$ appels à l'oracle pour f.

Preuve. Par des applications successives de la propriété de sous-modularité, nous obtenons pour $\emptyset \neq S \subseteq U$ (et pour $x \in S$) :

$$f(S) \;\le\; -f(\emptyset) + f(S\setminus\{x\}) + f(\{x\}) \;\le\; \cdots \;\le\; -|S| f(\emptyset) + f(\emptyset) + \sum_{x\in S} f(\{x\}),$$

et pour $S \subset U$ (et pour $y \in U \setminus S$) :

$$\begin{aligned} f(S) &\geq -f(\{y\}) + f(S \cup \{y\}) + f(\emptyset) \geq \cdots \\ &\geq -\sum_{y \in U \setminus S} f(\{y\}) + f(U) + |U \setminus S| f(\emptyset). \end{aligned}$$

□

Proposition 14.17. *Le problème suivant peut être résolu en temps polynomial : étant donné un ensemble fini U, une fonction sous-modulaire et monotone $f : 2^U \to \mathbb{Z}_+$ (donnée par un oracle) telle que $f(S) > 0$ pour $S \neq \emptyset$, un nombre $B \in \mathbb{N}$ tel que $f(S) \leq B$ pour tout $S \subseteq U$, et un vecteur $x \in \mathbb{Z}_+^U$, décider si $x \in P(f)$ et sinon retourner un ensemble $S \subseteq U$ tel que $\sum_{v \in S} x(v) > f(S)$.*

Preuve. C'est le PROBLÈME DE SÉPARATION appliqué au polymatroïde $P(f)$. Nous utiliserons le théorème 4.23, car nous avons déjà résolu le problème d'optimisation pour $P(f)$: l'ALGORITHME GLOUTON POUR LES POLYMATROÏDES maximise toute fonction linéaire sur $P(f)$ (théorème 14.11).

Nous devons vérifier les conditions du théorème 4.23. Comme le vecteur, dont toutes les composantes sont nulles, et les vecteurs unités sont tous dans $P(f)$, nous pouvons prendre $x_0 := \epsilon \mathbb{1}$ comme point à l'intérieur de $P(f)$, où $\epsilon = \frac{1}{|U|+1}$. Notons que $\text{taille}(x_0) = O(|U| \log |U|))$. De plus, chaque sommet de $P(f)$ est fourni par l'ALGORITHME GLOUTON POUR LES POLYMATROÏDES (pour une certaine fonction objectif ; voir théorème 14.11) et a ainsi pour taille $O(|U|(2 + \log B))$. Nous en concluons que le PROBLÈME DE SÉPARATION peut être résolu en temps polynomial. D'après le théorème 4.23, nous obtenons une inégalité induisant une facette de $P(f)$ qui est violée par x si $x \notin P(f)$. Cela correspond à un ensemble $S \subseteq U$ tel que $\sum_{v \in S} x(v) > f(S)$. □

Comme nous ne faisons pas l'hypothèse que f est monotone, nous ne pouvons pas appliquer ce résultat directement. À la place, nous considérons une autre fonction :

Proposition 14.18. *Soit $f : 2^U \to \mathbb{R}$ une fonction sous-modulaire et $\beta \in \mathbb{R}$. Alors $g : 2^U \to \mathbb{R}$, définie par*

$$g(X) := f(X) - \beta + \sum_{e \in X} (f(U \setminus \{e\}) - f(U)),$$

est sous-modulaire et monotone.

Preuve. La sous-modularité de g découle directement de la sous-modularité de f. Afin de montrer que g est monotone, considérons $X \subset U$ et $e \in U \setminus X$. On a $g(X \cup \{e\}) - g(X) = f(X \cup \{e\}) - f(X) + f(U \setminus \{e\}) - f(U) \geq 0$ puisque f est sous-modulaire. □

Théorème 14.19. *Le* PROBLÈME DE MINIMISATION D'UNE FONCTION SOUS-MODULAIRE *peut être résolu en temps polynomial en $|U| + \log \max\{|f(S)| : S \subseteq U\}$.*

Preuve. Soit U un ensemble fini ; supposons que f est donnée par un oracle. Calculons d'abord un nombre $B \in \mathbb{N}$ tel que $|f(S)| \leq B$ pour tout $S \subseteq U$ (voir proposition 14.16). Comme f est sous-modulaire, on a pour tout $e \in U$ et pour tout $X \subseteq U \setminus \{e\}$:

$$f(\{e\}) - f(\emptyset) \;\geq\; f(X \cup \{e\}) - f(X) \;\geq\; f(U) - f(U \setminus \{e\}). \tag{14.2}$$

Si, pour un $e \in U$, $f(\{e\}) - f(\emptyset) \leq 0$, alors, d'après (14.2), il existe un ensemble optimal S contenant e. Dans ce cas, nous considérons l'instance (U', B, f') définie par $U' := U \setminus \{e\}$ et $f'(X) := f(X \cup \{e\})$ pour $X \subseteq U \setminus \{e\}$, trouver un ensemble $S' \subseteq U'$ tel que $f'(S')$ soit minimum et fournir $S := S' \cup \{e\}$.

De façon similaire, si $f(U) - f(U \setminus \{e\}) \geq 0$, alors, d'après (14.2), il existe un ensemble optimum S ne contenant pas e. Dans ce cas, on minimise simplement la restriction de f à $U \setminus \{e\}$. Dans les deux cas, la taille de l'ensemble de référence est réduite.

Nous pouvons donc supposer que $f(\{e\}) - f(\emptyset) > 0$ et $f(U \setminus \{e\}) - f(U) > 0$ pour tout $e \in U$. Soit $x(e) := f(U \setminus \{e\}) - f(U)$. Pour chaque entier β tel que $-B \leq \beta \leq f(\emptyset)$, on définit $g(X) := f(X) - \beta + \sum_{e \in X} x(e)$. D'après la proposition 14.18, g est sous-modulaire et monotone. De plus, on a $g(\emptyset) = f(\emptyset) - \beta \geq 0$ et $g(\{e\}) = f(\{e\}) - \beta + x(e) > 0$ pour tout $e \in U$, et ainsi $g(X) > 0$ pour tout $\emptyset \neq X \subseteq U$. Nous appliquons maintenant la proposition 14.17 et vérifions si $x \in P(g)$. Si c'est le cas, on a $f(X) \geq \beta$ pour tout $X \subseteq U$ et nous avons fini. Sinon, nous obtenons un ensemble S tel que $f(S) < \beta$.

Nous appliquons une recherche dichotomique : en choisissant à chaque fois β de façon appropriée, on trouve, après $O(\log(2B))$ itérations, le nombre $\beta^* \in \{-B, -B+1, \ldots, f(\emptyset)\}$ pour lequel $f(X) \geq \beta^*$ pour tout $X \subseteq U$, mais $f(S) < \beta^* + 1$ pour un $S \subseteq U$. Cet ensemble S minimise f. □

Le premier algorithme fortement polynomial a été conçu par Grötschel, Lovász et Schrijver [1988], également fondé sur la méthode des ellipsoïdes. Des algorithmes combinatoires permettant de résoudre le PROBLÈME DE MINIMISATION D'UNE FONCTION SOUS-MODULAIRE en temps fortement polynomial ont été trouvés par Schrijver [2000] et indépendamment par Iwata, Fleischer et Fujishige [2001]. Nous décrivons l'algorithme de Schrijver à la section suivante.

14.4 Algorithme de Schrijver

Pour un ensemble fini U et une fonction sous-modulaire $f : 2^U \to \mathbb{R}$, supposons, sans perte de généralité, que $U = \{1, \ldots, n\}$ et $f(\emptyset) = 0$. L'ALGORITHME DE SCHRIJVER [2000] conserve, à chaque étape, un point x dans le polyèdre des bases de f, défini par

$$\left\{ x \in \mathbb{R}^U : \sum_{u \in A} x(u) \leq f(A) \text{ pour tout } A \subseteq U,\ \sum_{u \in U} x(u) = f(U) \right\}.$$

Mentionnons que l'ensemble des sommets de ce polyèdre des bases est précisément l'ensemble des vecteurs $b^{\prec}$ pour tous les ordres totaux $\prec$ de U, définis par

$$b^{\prec}(u) := f(\{v \in U : v \preceq u\}) - f(\{v \in U : v \prec u\})$$

($u \in U$). Ce résultat, dont nous n'aurons pas besoin ici, peut être prouvé de façon similaire au théorème 14.11 (exercice 13).

Le point x est toujours décrit par une combinaison convexe explicite $x = \lambda_1 b^{\prec_1} + \cdots + \lambda_k b^{\prec_k}$ de ces sommets. Initialement, on peut choisir $k = 1$ et n'importe quel ordre total.

ALGORITHME DE SCHRIJVER

Input Un ensemble fini $U = \{1, \ldots, n\}$. Une fonction sous-modulaire $f : 2^U \to \mathbb{Z}$ telle que $f(\emptyset) = 0$ (donnée par un oracle).

Output Un sous-ensemble $X \subseteq U$ tel que $f(X)$ soit minimum.

① Poser $k := 1$, considérer un ordre total $\prec_1$ sur U, et poser $x := b^{\prec_1}$.

② Poser $D := (U, A)$, où $A = \{(u, v) : u \prec_i v \text{ pour } i \in \{1, \ldots, k\}\}$.

③ Soient $P := \{v \in U : x(v) > 0\}$ et $N := \{v \in U : x(v) < 0\}$, et soit X l'ensemble des sommets non connectés à P dans D.
If $N \subseteq X$, **then stop else** notons $d(v)$ la distance de P à v dans D.

④ Choisir le sommet $t \in N$ connecté à P tel que $(d(t), t)$ soit lexicographiquement maximum.
Choisir le sommet maximal s tel que $(s, t) \in A$ et $d(s) = d(t) - 1$.
Soit $i \in \{1, \ldots, k\}$ tel que $\alpha := |\{v : s \prec_i v \preceq_i t\}|$ soit maximum (le nombre d'indices pour lesquels ce maximum est atteint sera noté β).

⑤ Soit $\prec_i^{s,u}$ l'ordre total obtenu en déplaçant u juste devant s dans l'ordre total $\prec_i$, et notons χ^u le vecteur d'incidence de u ($u \in U$).
Calculer un nombre ϵ tel que $0 \leq \epsilon \leq -x(t)$ et écrire $x' := x + \epsilon(\chi^t - \chi^s)$ comme combinaison convexe d'au plus n vecteurs, choisis parmi $b^{\prec_1}, \ldots, b^{\prec_k}$ et $b^{\prec_i^{s,u}}$ avec $s \prec_i u \preceq_i t$ et la condition supplémentaire que $b^{\prec_i}$ n'apparaît pas si $x'(t) < 0$.

⑥ Poser $x := x'$, renommer les vecteurs de la combinaison convexe de x par $b^{\prec_1}, \ldots, b^{\prec_k}$, et **go to** ②.

Théorème 14.20. (Schrijver [2000]) *L'*ALGORITHME DE SCHRIJVER *répond correctement.*

Preuve. L'algorithme se termine si D ne contient pas de chemin de P à N et fournit l'ensemble X de sommets non joignables depuis P. Il est clair que $N \subseteq X \subseteq U \setminus P$, donc $\sum_{u \in X} x(u) \leq \sum_{u \in W} x(u)$ pour tout $W \subseteq U$. De plus, il n'y a pas d'arc entrant en X ; donc, soit $X = \emptyset$, soit, pour tout $j \in \{1, \ldots, k\}$, il existe un $v \in X$ tel que $X = \{u \in U : u \preceq_j v\}$. Ainsi, par définition, $\sum_{u \in X} b^{\prec_j}(u) = f(X)$ pour tout $j \in \{1, \ldots, k\}$. De plus, d'après la proposition

14.10, $\sum_{u\in W} b^{\prec_j}(u) \le f(W)$ pour tout $W \subseteq U$ et $j \in \{1,\dots,k\}$. Ainsi, pour tout $W \subseteq U$,

$$\begin{aligned} f(W) &\ge \sum_{j=1}^{k} \lambda_j \sum_{u\in W} b^{\prec_j}(u) = \sum_{u\in W}\sum_{j=1}^{k} \lambda_j b^{\prec_j}(u) = \sum_{u\in W} x(u) \\ &\ge \sum_{u\in X} x(u) = \sum_{u\in X}\sum_{j=1}^{k} \lambda_j b^{\prec_j}(u) = \sum_{j=1}^{k} \lambda_j \sum_{u\in X} b^{\prec_j}(u) = f(X), \end{aligned}$$

ce qui prouve que X est une solution optimale. □

Lemme 14.21. (Schrijver [2000]) *Chaque itération peut être exécutée en temps $O(n^3 + \gamma n^2)$, où γ est le temps d'un appel à l'oracle.*

Preuve. Il suffit de montrer que l'étape ⑤ peut être effectuée en temps $O(n^3 + \gamma n^2)$. Soit $x = \lambda_1 b^{\prec_1} + \cdots + \lambda_k b^{\prec_k}$ et $s \prec_i t$. Nous montrons d'abord :
Affirmation : $\delta(\chi^t - \chi^s)$, pour $\delta \ge 0$, peut s'écrire comme combinaison convexe des vecteurs $b^{\prec_i^{s,v}} - b^{\prec_i}$ pour $s \prec_i v \preceq_i t$ en temps $O(\gamma n^2)$.

Afin de prouver cela, nous avons besoin de quelques résultats préliminaires. Soit $s \prec_i v \preceq_i t$. Par définition, $b^{\prec_i^{s,v}}(u) = b^{\prec_i}(u)$ pour $u \prec_i s$ ou $u \succ_i v$. Comme f est sous-modulaire, on a pour $s \preceq_i u \prec_i v$:

$$\begin{aligned} b^{\prec_i^{s,v}}(u) &= f(\{w \in U : w \preceq_i^{s,v} u\}) - f(\{w \in U : w \prec_i^{s,v} u\}) \\ &\le f(\{w \in U : w \preceq_i u\}) - f(\{w \in U : w \prec_i u\}) = b^{\prec_i}(u). \end{aligned}$$

De plus, pour $u = v$, on a :

$$\begin{aligned} b^{\prec_i^{s,v}}(v) &= f(\{w \in U : w \preceq_i^{s,v} v\}) - f(\{w \in U : w \prec_i^{s,v} v\}) \\ &= f(\{w \in U : w \prec_i s\} \cup \{v\}) - f(\{w \in U : w \prec_i s\}) \\ &\ge f(\{w \in U : w \preceq_i v\}) - f(\{w \in U : w \prec_i v\}) \\ &= b^{\prec_i}(v). \end{aligned}$$

Finalement, observons que $\sum_{u\in U} b^{\prec_i^{s,v}}(u) = f(U) = \sum_{u\in U} b^{\prec_i}(u)$.

L'affirmation étant évidente si $b^{\prec_i^{s,v}} = b^{\prec_i}$ pour un $s \prec_i v \preceq_i t$, on peut supposer $b^{\prec_i^{s,v}}(v) > b^{\prec_i}(v)$ pour tout $s \prec_i v \preceq_i t$. Nous posons récursivement

$$\kappa_v := \frac{\chi^t_v - \sum_{v \prec_i w \preceq_i t} \kappa_w (b^{\prec_i^{s,w}}(v) - b^{\prec_i}(v))}{b^{\prec_i^{s,v}}(v) - b^{\prec_i}(v)} \ge 0$$

pour $s \prec_i v \preceq_i t$, et obtenons $\sum_{s\prec_i v\preceq_i t} \kappa_v (b^{\prec_i^{s,v}} - b^{\prec_i}) = \chi^t - \chi^s$, car $\sum_{s\prec_i v\preceq_i t} \kappa_v (b^{\prec_i^{s,v}}(u) - b^{\prec_i}(u)) = \sum_{u\preceq_i v\preceq_i t} \kappa_v (b^{\prec_i^{s,v}}(u) - b^{\prec_i}(u)) = \chi^t_u$ pour tout $s \prec_i u \preceq_i t$, et la somme sur toutes les composantes est nulle.

En posant $\delta := \frac{1}{\sum_{s\prec_i v\preceq_i t} \kappa_v}$ et en multipliant chaque κ_u par δ, l'affirmation est démontrée.

Considérons maintenant $\epsilon := \min\{\lambda_i\delta, -x(t)\}$ et $x' := x + \epsilon(\chi^t - \chi^s)$. Si $\epsilon = \lambda_i\delta \leq -x(t)$, on a alors $x' = \sum_{j=1}^{k} \lambda_j b^{\prec_j} + \lambda_i \sum_{s \prec_i v \preceq_i t} \kappa_v (b^{\prec_i^{s,v}} - b^{\prec_i})$, c.-à-d. que l'on a écrit x' comme combinaison convexe des $b^{\prec_j}$ ($j \in \{1, \ldots, k\} \setminus \{i\}$) et des $b^{\prec_i^{s,v}}$ ($s \prec_i v \preceq_i t$). Si $\epsilon = -x(t)$, on peut en plus utiliser $b^{\prec_i}$ dans la combinaison convexe.

Finalement, nous réduisons cette combinaison convexe à au plus n vecteurs en temps $O(n^3)$, de la même façon qu'à l'exercice 5 du chapitre 4. □

Lemme 14.22. (Vygen [2003]) *L'*ALGORITHME DE SCHRIJVER *se termine après* $O(n^5)$ *itérations.*

Preuve. Si un arc (v, w) est introduit, après qu'un nouveau vecteur $b^{\prec_i^{s,v}}$ a été ajouté à l'étape ⑤ d'une itération, alors $s \preceq_i w \prec_i v \preceq_i t$ à cette itération. Ainsi $d(w) \leq d(s) + 1 = d(t) \leq d(v) + 1$ à cette itération, et l'introduction du nouvel arc ne peut pas réduire la distance de P à tout $v \in U$. Comme l'étape ⑤ garantit qu'aucun élément n'est ajouté à P, la distance $d(v)$ ne diminue jamais pour tout $v \in U$.

Appelons *bloc* une suite d'itérations où le couple (s, t) n'est pas modifié. Remarquons que chaque bloc correspond à $O(n^2)$ itérations, car (α, β) décroît lexicographiquement à chaque itération associée à un bloc. Il reste à prouver qu'il y a $O(n^3)$ blocs.

Un bloc se termine seulement pour l'une des raisons suivantes (par le choix de t et s, puisqu'une itération telle que $t = t^*$ n'ajoute pas d'arc dont l'extrémité finale soit t^*, et puisqu'un sommet v peut être ajouté à N seulement si $v = s$ et par conséquent $d(v) < d(t)$) :

(a) La distance $d(v)$ augmente pour un $v \in U$.

(b) t est enlevé de N.

(c) (s, t) est enlevé de A.

Nous comptons maintenant le nombre de blocs de chacun de ces trois types. Il y a évidemment $O(n^2)$ blocs du type (a).

Considérons maintenant les blocs du type (b). Nous affirmons que, pour chaque $t^* \in U$, il y a $O(n^2)$ itérations telles que $t = t^*$ et $x'(t) = 0$. Cela est facile à voir : entre deux telles itérations, $d(v)$ doit changer pour un $v \in U$, et cela peut se produire $O(n^2)$ fois puisque les valeurs de d ne peuvent qu'augmenter. Il y a ainsi $O(n^3)$ blocs du type (b).

Finalement, nous montrons qu'il y a $O(n^3)$ blocs du type (c). Il suffit de montrer que $d(t)$ sera modifié avant que l'on obtienne le bloc suivant associé au même couple (s, t).

Pour $s, t \in U$, nous dirons que s est *t-gênant* si $(s, t) \notin A$ ou $d(t) \leq d(s)$. Prenons $s^*, t^* \in U$, et considérons la période de temps qui commence après un bloc tel que $s = s^*$ et $t = t^*$ se terminant, car (s^*, t^*) est supprimé de A, et qui dure jusqu'au changement suivant de $d(t^*)$. Nous prouvons que chaque $v \in \{s^*, \ldots, n\}$ est t^*-gênant pendant cette période. L'application de ce résultat avec $v = s^*$ termine la preuve.

Au début de cette période, chaque $v \in \{s^*+1, \ldots, n\}$ est t^*-gênant en raison du choix de $s = s^*$ à l'itération qui précéde immédiatement la période considérée. s^* est aussi t^*-gênant, car (s^*, t^*) est supprimé de A. Comme $d(t^*)$ reste constant au cours de la période de temps considérée et que $d(v)$ ne diminue jamais pour aucun v, nous avons seulement à vérifier ce qui se passe lors de l'introduction de nouveaux arcs.

Supposons que, pour un élément $v \in \{s^*, \ldots, n\}$, l'arc (v, t^*) soit ajouté à A après une itération qui sélectionne le couple (s, t). Alors, d'après les remarques du début de cette preuve, $s \preceq_i t^* \prec_i v \preceq_i t$ à cette itération, et ainsi $d(t^*) \leq d(s) + 1 = d(t) \leq d(v) + 1$. Nous distinguons maintenant deux cas : si $s > v$, alors on a $d(t^*) \leq d(s)$: soit parce que $t^* = s$, soit parce que s était t^*-gênant et $(s, t^*) \in A$. Si $s < v$, alors on a $d(t) \leq d(v)$: soit parce que $t = v$, soit par le choix s et puisque $(v, t) \in A$. Dans les deux cas, nous concluons que $d(t^*) \leq d(v)$, et v reste t^*-gênant. □

Le théorème 14.20, le lemme 14.21 et le lemme 14.22 impliquent :

Théorème 14.23. *Le* PROBLÈME DE MINIMISATION D'UNE FONCTION SOUS-MODULAIRE *peut être résolu en temps $O(n^8 + \gamma n^7)$, où γ correspond au temps d'un appel à un oracle.* □

Iwata [2002] a décrit un algorithme entièrement combinatoire (utilisant seulement des additions, soustractions, comparaisons et des appels à un oracle, mais pas de multiplication ou de division). Il a ensuite amélioré le temps de calcul (Iwata [2003]). L'algorithme fortement polynomial actuellement le plus rapide a été trouvé par Orlin [2007] ; son temps de calcul est en $O(n^6 + \gamma n^5)$.

14.5 Fonctions sous-modulaires symétriques

Une fonction sous-modulaire $f : 2^U \to \mathbb{R}$ est dite **symétrique** si $f(A) = f(U \setminus A)$ pour tout $A \subseteq U$. Dans ce cas particulier, le PROBLÈME DE MINIMISATION D'UNE FONCTION SOUS-MODULAIRE est trivial, puisque $2f(\emptyset) = f(\emptyset) + f(U) \leq f(A) + f(U \setminus A) = 2f(A)$ pour tout $A \subseteq U$, ce qui implique que l'ensemble vide est optimal. Ainsi, le problème est intéressant seulement si ce cas trivial est exclu : on recherche un sous-ensemble propre non vide A de U tel que $f(A)$ soit minimum.

En généralisant l'algorithme du paragraphe 8.7, Queyranne [1998] a trouvé un algorithme combinatoire relativement simple résolvant ce problème, qui requiert seulement $O(n^3)$ appels à un oracle. Le lemme suivant est une généralisation du lemme 8.38 (exercice 14) :

Lemme 14.24. *Étant donné une fonction sous-modulaire symétrique $f : 2^U \to \mathbb{R}$ avec $n := |U| \geq 2$, on peut trouver deux éléments $x, y \in U$ tels que $x \neq y$ et $f(\{x\}) = \min\{f(X) : x \in X \subseteq U \setminus \{y\}\}$ en temps $O(n^2\theta)$, où θ est une borne sur le temps d'un appel à l'oracle associé à f.*

Preuve. On construit un ordre $U = \{u_1, \ldots, u_n\}$ en suivant la procédure suivante pour $k = 1, \ldots, n-1$. Supposons que $u_1, \ldots, u_{k-1}$ sont déjà construits ; considérons $U_{k-1} := \{u_1, \ldots, u_{k-1}\}$. Pour $C \subseteq U$, on définit

$$w_k(C) \;:=\; f(C) - \frac{1}{2}(f(C \setminus U_{k-1}) + f(C \cup U_{k-1}) - f(U_{k-1})).$$

Remarquons que w_k est également symétrique. Soit u_k un élément de $U \setminus U_{k-1}$ maximisant $w_k(\{u_k\})$.

Finalement, soit u_n le seul élément de $U \setminus \{u_1, \ldots, u_{n-1}\}$. Évidemment, la construction de l'ordre $u_1, \ldots, u_n$ peut se faire en temps $O(n^2\theta)$.

Affirmation : pour tout $k = 1, \ldots, n-1$ et tout $x, y \in U \setminus U_{k-1}$ tel que $x \neq y$ et $w_k(\{x\}) \leq w_k(\{y\})$, on a

$$w_k(\{x\}) \;=\; \min\{w_k(C) : x \in C \subseteq U \setminus \{y\}\}.$$

Nous prouvons cette affirmation par induction sur k. Pour $k = 1$ l'affirmation est évidente puisque $w_1(C) = \frac{1}{2}f(\emptyset)$ pour tout $C \subseteq U$.

Supposons maintenant que $k > 1$ et que $x, y \in U \setminus U_{k-1}$ tel que $x \neq y$ et $w_k(\{x\}) \leq w_k(\{y\})$. De plus, soit $Z \subseteq U$ tel que $u_{k-1} \notin Z$, et soit $z \in Z \setminus U_{k-1}$. Par le choix de u_{k-1}, on a $w_{k-1}(\{z\}) \leq w_{k-1}(\{u_{k-1}\})$; ainsi par l'hypothèse d'induction, on obtient $w_{k-1}(\{z\}) \leq w_{k-1}(Z)$. De plus, la sous-modularité de f implique

$$\begin{aligned}
& (w_k(Z) - w_{k-1}(Z)) - (w_k(\{z\}) - w_{k-1}(\{z\})) \\
= \;& \frac{1}{2}\left(f(Z \cup U_{k-2}) - f(Z \cup U_{k-1}) - f(U_{k-2}) + f(U_{k-1})\right) \\
& - \frac{1}{2}\left(f(\{z\} \cup U_{k-2}) - f(\{z\} \cup U_{k-1}) - f(U_{k-2}) + f(U_{k-1})\right) \\
= \;& \frac{1}{2}(f(Z \cup U_{k-2}) + f(\{z\} \cup U_{k-1}) - f(Z \cup U_{k-1}) - f(\{z\} \cup U_{k-2})) \\
\geq \;& 0.
\end{aligned}$$

Ainsi $w_k(Z) - w_k(\{z\}) \geq w_{k-1}(Z) - w_{k-1}(\{z\}) \geq 0$.

Pour finir la preuve de cette affirmation, considérons $C \subseteq U$ tel que $x \in C$ et $y \notin C$. Il y a alors deux cas :

Cas 1 : $u_{k-1} \notin C$. Alors le résultat précédent appliqué à $Z = C$ et $z = x$ donne $w_k(C) \geq w_k(\{x\})$ comme voulu.

Cas 2 : $u_{k-1} \in C$. Alors nous appliquons le résultat précédent à $Z = U \setminus C$ et $z = y$ et obtenons $w_k(C) = w_k(U \setminus C) \geq w_k(\{y\}) \geq w_k(\{x\})$.

Cela termine la preuve de l'affirmation. En l'appliquant à $k = n-1$, $x = u_n$ et $y = u_{n-1}$, on obtient

$$w_{n-1}(\{u_n\}) \;=\; \min\{w_{n-1}(C) : u_n \in C \subseteq U \setminus \{u_{n-1}\}\}.$$

Comme $w_{n-1}(C) = f(C) - \frac{1}{2}(f(\{u_n\}) + f(U \setminus \{u_{n-1}\}) - f(U_{n-2}))$ pour tout $C \subseteq U$ tel que $u_n \in C$ et $u_{n-1} \notin C$, le lemme est prouvé (prendre $x := u_n$ et $y := u_{n-1}$). □

La preuve précédente est due à Fujishige [1998]. On peut maintenant procéder de façon analogue à la preuve du théorème 8.39 :

Théorème 14.25. (Queyranne [1998]) *Étant donné une fonction sous-modulaire symétrique $f : 2^U \to \mathbb{R}$, un sous-ensemble propre non vide A de U tel que $f(A)$ soit minimum peut être trouvé en temps $O(n^3\theta)$ où θ est une borne sur le temps d'un appel à l'oracle associé à f.*

Preuve. Si $|U| = 1$, le problème est trivial. Sinon, on applique le lemme 14.24 et on trouve deux éléments $x, y \in U$ tels que $f(\{x\}) = \min\{f(X) : x \in X \subseteq U \setminus \{y\}\}$ en temps $O(n^2\theta)$. Ensuite, on trouve récursivement un sous-ensemble propre non vide de $U \setminus \{x\}$ minimisant la fonction $f' : 2^{U\setminus\{x\}} \to \mathbb{R}$, définie par $f'(X) := f(X)$ si $y \notin X$ et $f'(X) := f(X \cup \{x\})$ si $y \in X$. On peut déjà observer que f' est symétrique et sous-modulaire.

Soit $\emptyset \neq Y \subset U \setminus \{x\}$ un ensemble minimisant f' ; on peut supposer sans perte de généralité que $y \in Y$ (car f' est symétrique). Nous affirmons que soit $\{x\}$, soit $Y \cup \{x\}$ minimise f (sur tous les sous-ensembles propres non vides de U). Pour prouver cela, considérons un sous-ensemble $C \subset U$ tel que $x \in C$. Si $y \notin C$, alors on a $f(\{x\}) \leq f(C)$ d'après le choix de x et y. Si $y \in C$, alors $f(C) = f'(C \setminus \{x\}) \geq f'(Y) = f(Y \cup \{x\})$. Ainsi $f(C) \geq \min\{f(\{x\}), f(Y \cup \{x\})\}$ pour tous les sous-ensembles propres non vides C de U.

Pour obtenir le temps de calcul annoncé, on ne peut bien sûr pas calculer f' explicitement. Nous stockons plutôt une partition de U, constituée initialement de tous les singletons. À chaque étape de la récursion, on forme l'union des deux ensembles de la partition qui contiennent x et y. De cette façon, f' peut être calculée efficacement (en utilisant l'oracle associé à f). □

Ce résultat a, par la suite, été généralisé par Nagamochi et Ibaraki [1998] et par Rizzi [2000].

Exercices

1. Soit G un graphe non orienté et M un couplage maximum de G. Soit $\mathcal{F}$ la famille des sous-ensembles $X \subseteq E(G)$ pour lesquels il existe une forêt spéciale de blossoms F relativement à M telle que $E(F)\backslash M = X$. Prouver que $(E(G)\backslash M, \mathcal{F})$ est un greedoïde.
   Indication : utiliser l'exercice 23 du chapitre 10.
2. Soit $(E, \mathcal{F})$ un greedoïde et $c' : E \to \mathbb{R}_+$. Nous considérons la fonction seuil $c(F) := \min\{c'(e) : e \in F\}$ pour $F \subseteq E$. Montrer que l'ALGORITHME GLOUTON POUR LES GREEDOÏDES, lorsqu'il est appliqué à $(E, \mathcal{F})$ et c, permet de trouver un ensemble $F \in \mathcal{F}$ tel que $c(F)$ soit maximum.
3. Cet exercice montre que les greedoïdes peuvent aussi être définis comme des langages (voir définition 15.1). Soit E un ensemble fini. Un langage L sur l'alphabet E est appelé un langage de greedoïde si :

(a) L contient la chaîne vide.

(b) $x_i \neq x_j$ pour tout $(x_1, \ldots, x_n) \in L$ et $1 \leq i < j \leq n$.

(c) $(x_1, \ldots, x_{n-1}) \in L$ pour tout $(x_1, \ldots, x_n) \in L$.

(d) Si $(x_1, \ldots, x_n), (y_1, \ldots, y_m) \in L$ avec $m < n$, alors il existe un $i \in \{1, \ldots, n\}$ tel que $(y_1, \ldots, y_m, x_i) \in L$.

L est appelé un langage d'antimatroïde s'il vérifie (a), (b), (c) et

(d′) Si $(x_1, \ldots, x_n), (y_1, \ldots, y_m) \in L$ avec $\{x_1, \ldots, x_n\} \not\subseteq \{y_1, \ldots, y_m\}$, alors il existe un $i \in \{1, \ldots, n\}$ tel que $(y_1, \ldots, y_m, x_i) \in L$.

Prouver : Un langage L sur l'alphabet E est un langage de greedoïde (un langage d'antimatroïde) si et seulement si le système d'ensemble $(E, \mathcal{F})$ est un greedoïde (antimatroïde), où $\mathcal{F} := \{\{x_1, \ldots, x_n\} : (x_1, \ldots, x_n) \in L\}$.

4. Soit U un ensemble fini et $f : 2^U \to \mathbb{R}$. Prouver que f est sous-modulaire si et seulement si $f(X \cup \{y, z\}) - f(X \cup \{y\}) \leq f(X \cup \{z\} - f(X)$ pour tout $X \subseteq U$ et $y, z \in U$.

5. Soit P un polymatroïde non vide. Montrer qu'il existe alors une fonction monotone et sous-modulaire f telle que $f(\emptyset) = 0$ et $P = P(f)$. ($f : 2^E \to \mathbb{R}$ est dite monotone si $f(A) \leq f(B)$ pour tout $A \subseteq B \subseteq E$).

* 6. Prouver qu'un ensemble compact non vide $P \subseteq \mathbb{R}^n_+$ est un polymatroïde si et seulement si :

(a) Pour tout $0 \leq x \leq y \in P$, on a $x \in P$.

(b) Pour tout $x \in \mathbb{R}^n_+$ et tout $y, z \leq x$ tels que $y, z \in P$ et maximaux pour cette propriété (c.-à-d. $y \leq w \leq x$ et $w \in P$ implique $w = y$, et $z \leq w \leq x$ et $w \in P$ implique $w = z$) on a $\mathbb{1}y = \mathbb{1}z$.

Note : cela est la définition initiale donnée par Edmonds [1970].

7. Prouver que l'ALGORITHME GLOUTON POUR LES POLYMATROÏDES, lorsqu'il est appliqué à un vecteur $c \in \mathbb{R}^E_+$ et à une fonction sous-modulaire, mais pas nécessairement monotone $f : 2^E \to \mathbb{R}$ telle que $f(\emptyset) \geq 0$, permet de trouver

$$\max\{cx : \sum_{e \in A} x_e \leq f(A) \text{ pour tout } A \subseteq E\}.$$

8. Prouver le théorème 14.12 dans le cas particulier où f et g sont des fonctions rang de matroïdes, en construisant une solution duale optimale entière à partir de c_1 et c_2 générés par l'ALGORITHME DE L'INTERSECTION DE MATROÏDES AVEC POIDS.
(Frank [1981])

* 9. Soit S un ensemble fini et $f : 2^S \to \mathbb{R}$. Définissons $f' : \mathbb{R}^S_+ \to \mathbb{R}$ de la façon suivante. Pour tout $x \in \mathbb{R}^S_+$, il existe une écriture unique de x sous la forme $x = \sum_{i=1}^k \lambda_i \chi^{T_i}$ où $k \in \mathbb{Z}_+$, $\lambda_1, \ldots, \lambda_k > 0$ et $\emptyset \subset T_1 \subset T_2 \subset \cdots \subset T_k \subseteq S$ et où χ^{T_i} est le vecteur d'incidence de T_i. Alors $f'(x) := \sum_{i=1}^k \lambda_i f(T_i)$.
Prouver que f est sous-modulaire si et seulement si f' est convexe.
(Lovász [1983])

10. Soit E un ensemble fini et $f : 2^E \to \mathbb{R}_+$ une fonction sous-modulaire telle que $f(\{e\}) \leq 2$ pour tout $e \in E$. (Le couple (E, f) est parfois appelé un 2-polymatroïde.) Le PROBLÈME DU COUPLAGE DANS UN POLYMATROÏDE recherche un ensemble de cardinalité maximale $X \subseteq E$ tel que $f(X) = 2|X|$. (f est bien entendu donnée par un oracle.)

    Soient $E_1, \ldots, E_k$ des paires non ordonnées deux à deux disjointes et soit $(E, \mathcal{F})$ un matroïde (donné par un oracle d'indépendance), où $E = E_1 \cup \cdots \cup E_k$. Le PROBLÈME DE PARITÉ D'UN MATROÏDE recherche un ensemble de cardinalité maximale $I \subseteq \{1, \ldots, k\}$ tel que $\bigcup_{i \in I} E_i \in \mathcal{F}$.

    (a) Montrer que le PROBLÈME DE PARITÉ D'UN MATROÏDE se réduit polynomialement au PROBLÈME DU COUPLAGE DANS UN POLYMATROÏDE.

    $*$ (b) Montrer que le PROBLÈME DU COUPLAGE DANS UN POLYMATROÏDE se réduit polynomialement au PROBLÈME DE PARITÉ D'UN MATROÏDE.
    Indication : utiliser un algorithme qui résout le PROBLÈME DE MINIMISATION D'UNE FONCTION SOUS-MODULAIRE.

    $*$ (c) Montrer qu'il n'existe pas d'algorithme pour le PROBLÈME DU COUPLAGE DANS UN POLYMATROÏDE dont le temps de calcul soit polynomial en $|E|$. (Jensen et Korte [1982], Lovász [1981])

    (Un problème se réduit polynomialement à un autre s'il peut être résolu à l'aide d'un algorithme polynomial fondé sur un oracle qui fournit la solution de l'autre problème ; voir chapitre 15.)
    Note : un algorithme polynomial a été donné par Lovász [1980,1981] pour un cas particulier important.

11. Une fonction $f : 2^S \to \mathbb{R} \cup \{\infty\}$ est dite sous-modulaire croisante si $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ pour tout couple d'ensembles $X, Y \subseteq S$ tels que $X \cap Y \neq \emptyset$ et $X \cup Y \neq S$.

    Le PROBLÈME DU FLOT SOUS-MODULAIRE est le suivant : étant donné un graphe orienté G, des fonctions $l : E(G) \to \mathbb{R} \cup \{-\infty\}$, $u : E(G) \to \mathbb{R} \cup \{\infty\}$, $c : E(G) \to \mathbb{R}$, et une fonction sous-modulaire croisante $b : 2^{V(G)} \to \mathbb{R} \cup \{\infty\}$. Alors un flot sous-modulaire réalisable est une fonction $f : E(G) \to \mathbb{R}$ telle que $l(e) \leq f(e) \leq u(e)$ pour tout $e \in E(G)$ et

    $$\sum_{e \in \delta^-(X)} f(e) - \sum_{e \in \delta^+(X)} f(e) \;\leq\; b(X)$$

    pour tout $X \subseteq V(G)$. L'objectif est de décider s'il existe un flot réalisable et, si oui, d'en trouver un dont le coût $\sum_{e \in E(G)} c(e) f(e)$ soit minimum.

    Montrer que ce problème généralise le PROBLÈME DU FLOT DE COÛT MINIMUM et le problème de l'optimisation d'une fonction linéaire sur l'intersection de deux polymatroïdes.
    Note : le PROBLÈME DU FLOT SOUS-MODULAIRE, introduit par Edmonds et Giles [1977], peut être résolu en temps fortement polynomial ; voir Fujishige, Röck et Zimmermann [1989]. Voir aussi Fleischer et Iwata [2000].

∗ 12. Montrer que le système d'inégalités de la description d'un flot sous-modulaire réalisable (exercice 11) est TDI. Montrer que cela implique les théorèmes 14.12 et 19.10.
(Edmonds et Giles [1977])

13. Prouver que l'ensemble des sommets du polyèdre des bases est précisément l'ensemble des vecteurs $b^{\prec}$ pour tous les ordres totaux $\prec$ de U, où

$$b^{\prec}(u) := f(\{v \in U : v \preceq u\}) - f(\{v \in U : v \prec u\})$$

($u \in U$).
Indication : voir la preuve du théorème 14.11.

14. Montrer que le lemme 8.38 est un cas particulier du lemme 14.24.

Références

Littérature générale :

Bixby, R.E., Cunningham, W.H. [1995] : Matroid optimization and algorithms. In : Handbook of Combinatorics ; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, eds.), Elsevier, Amsterdam, 1995

Björner, A., Ziegler, G.M. [1992] : Introduction to greedoids. In : Matroid Applications (N. White, ed.), Cambridge University Press, Cambridge 1992

Fujishige, S. [2005] : Submodular Functions and Optimization. Second Edition. Elsevier, Amsterdam 2005

Korte, B., Lovász, L., Schrader, R. [1991] : Greedoids. Springer, Berlin 1991

McCormick, S.T. [2004] : Submodular function minimization. In : Discrete Optimization (K. Aardal, G.L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam 2005

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 44–49

Références citées :

Edmonds, J. [1970] : Submodular functions, matroids and certain polyhedra. In : Combinatorial Structures and Their Applications ; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schonheim, eds.), Gordon and Breach, New York 1970, pp. 69–87

Edmonds, J. [1979] : Matroid intersection. In : Discrete Optimization I ; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, eds.), North-Holland, Amsterdam 1979, pp. 39–49

Edmonds, J., Giles, R. [1977] : A min-max relation for submodular functions on graphs. In : Studies in Integer Programming ; Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, eds.), North-Holland, Amsterdam 1977, pp. 185–204

Fleischer, L., Iwata, S. [2000] : Improved algorithms for submodular function minimization and submodular flow. Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (2000), 107–116

Frank, A. [1981] : A weighted matroid intersection algorithm. Journal of Algorithms 2 (1981), 328–336

Frank, A. [1982] : An algorithm for submodular functions on graphs. In : Bonn Workshop on Combinatorial Optimization ; Annals of Discrete Mathematics 16 (A. Bachem, M. Grötschel, B. Korte, eds.), North-Holland, Amsterdam 1982, pp. 97–120

Fujishige, S. [1998] : Another simple proof of the validity of Nagamochi and Ibaraki's min-cut algorithm and Queyranne's extension to symmetric submodular function minimization. Journal of the Operations Research Society of Japan 41 (1998), 626–628

Fujishige, S., Röck, H., Zimmermann, U. [1989] : A strongly polynomial algorithm for minimum cost submodular flow problems. Mathematics of Operations Research 14 (1989), 60–69

Grötschel, M., Lovász, L., Schrijver, A. [1981] : The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1 (1981), 169–197

Grötschel, M., Lovász, L., Schrijver, A. [1988] : Geometric Algorithms and Combinatorial Optimization. Springer, Berlin 1988

Iwata, S. [2002] : A fully combinatorial algorithm for submodular function minimization. Journal of Combinatorial Theory B 84 (2002), 203–212

Iwata, S. [2003] : A faster scaling algorithm for minimizing submodular functions. SIAM Journal on Computing 32 (2003), 833–840

Iwata, S., Fleischer, L., L., Fujishige, S. [2001] : A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. Journal of the ACM 48 (2001), 761–777

Jensen, P.M., Korte, B. [1982] : Complexity of matroid property algorithms. SIAM Journal on Computing 11 (1982), 184–190

Lovász, L. [1980] : Matroid matching and some applications. Journal of Combinatorial Theory B 28 (1980), 208–236

Lovász, L. [1981] : The matroid matching problem. In : Algebraic Methods in Graph Theory ; Vol. II (L. Lovász, V.T. Sós, eds.), North-Holland, Amsterdam 1981, 495–517

Lovász, L. [1983] : Submodular functions and convexity. In : Mathematical Programming : The State of the Art – Bonn 1982 (A. Bachem, M. Grötschel, B. Korte, eds.), Springer, Berlin 1983

Nagamochi, H., Ibaraki, T. [1998] : A note on minimizing submodular functions. Information Processing Letters 67 (1998), 239–244

Orlin, J.B. [2007] : A faster strongly polynomial time algorithm for submodular function minimization. In : Integer Programming and Combinatorial Optimization ; Proceedings of the 12th International IPCO Conference ; LNCS 4513 (M. Fischetti, D.P. Williamson, eds.), Springer, Berlin 2007, pp. 240–251

Queyranne, M. [1998] : Minimizing symmetric submodular functions. Mathematical Programming B 82 (1998), 3–12

Rizzi, R. [2000] : On minimizing symmetric set functions. Combinatorica 20 (2000), 445–450

Schrijver, A. [2000] : A combinatorial algorithm minimizing submodular functions in strongly polynomial time. Journal of Combinatorial Theory B 80 (2000), 346–355

Vygen, J. [2003] : A note on Schrijver's submodular function minimization algorithm. Journal of Combinatorial Theory B 88 (2003), 399–402

Chapitre 15

NP-complétude

Pour de nombreux problèmes d'optimisation combinatoire, on connaît un algorithme polynomial ; les plus importants sont présentés dans ce livre. Cependant, il existe aussi de nombreux problèmes pour lesquels on ne connaît pas d'algorithme polynomial. On ne peut pas prouver qu'il n'en existe pas, mais on peut cependant montrer que l'existence d'un algorithme polynomial pour un problème «difficile» (plus précisément : *NP*-difficile) impliquerait l'existence d'un algorithme polynomial pour presque tous les problèmes présentés dans ce livre (plus précisément : tous les problèmes *NP*-faciles).

Afin de formaliser ce concept et de prouver l'affirmation précédente, nous avons besoin d'un modèle de machine, c.-à-d. d'une définition précise d'un algorithme polynomial. Nous présentons ainsi les machines de Turing au paragraphe 15.1. Ce modèle théorique n'est pas adapté à la description d'algorithmes plus compliqués. Cependant, nous montrerons qu'il est équivalent à notre description informelle des algorithmes : chaque algorithme de ce livre peut, en théorie, être décrit par une machine de Turing, avec une perte d'efficacité polynomialement bornée. Nous précisons cela au paragraphe 15.2.

Au paragraphe 15.3, nous introduisons les problèmes de décision, et en particulier les classes *P* et *NP*. Alors que la classe *NP* contient la plupart des problèmes de décision figurant dans ce livre, la classe *P* contient uniquement ceux pour lesquels il existe des algorithmes polynomiaux. Savoir si $P = NP$ est une question ouverte. Bien que nous présentions de nombreux problèmes dans *NP* pour lesquels on ne connaît pas d'algorithme polynomial, personne n'a pu prouver (jusqu'à maintenant) qu'il n'en existe pas. Nous précisons ce que signifie pouvoir réduire un problème à un autre ou dire qu'un problème est au moins aussi difficile qu'un autre. En particulier, les problèmes les plus difficiles de la classe *NP* sont les problèmes *NP*-complets ; ils peuvent être résolus en temps polynomial si et seulement si $P = NP$.

Au paragraphe 15.4, nous présentons le premier problème *NP*-complet, appelé SATISFAISABILITÉ. Au paragraphe 15.5, nous prouvons que d'autres problèmes de décision, plus étroitement liés à l'optimisation combinatoire, sont *NP*-complets.

Aux paragraphes 15.6 et 15.7, nous présentons des notions étroitement liées aux précédentes, qui s'étendent également aux problèmes d'optimisation.

15.1 Machines de Turing

Dans ce paragraphe, nous introduisons un modèle de calcul très simple : la machine de Turing. Ce modèle est une suite d'instructions simples s'exécutant sur une chaîne (de caractères). L'input et l'output seront une chaîne binaire :

Définition 15.1. *Un* **alphabet** *est un ensemble fini d'au moins deux éléments, ne contenant pas le symbole particulier* $\sqcup$ *(que nous utiliserons pour les blancs). Pour un alphabet* A*, nous notons par une* **chaîne** *sur* A*, une suite finie d'éléments de* A*, par* A^n *l'ensemble des chaînes de longueur* n*, et par* $A^* := \bigcup_{n\in\mathbb{Z}_+} A^n$ *l'ensemble de toutes les chaînes sur* A*. Nous utilisons la convention que* A^0 *contient exactement un élément, la* **chaîne vide***. Un* **langage** *sur* A *est un sous-ensemble de* A^**. Les éléments d'un langage sont souvent appelés des* **mots***. Si* $x \in A^n$*, nous notons* $\text{taille}(x) := n$ *la* **longueur** *de la chaîne.*

Nous utiliserons souvent l'alphabet $A = \{0, 1\}$ et l'ensemble $\{0, 1\}^*$ de toutes les chaînes en 0-1 (ou **chaînes binaires**). Les composantes d'une chaîne en 0-1 sont parfois appelées bits. Il y a donc exactement une seule chaîne en 0-1 de longueur zéro, la chaîne vide. Un langage sur $\{0, 1\}$ est un sous-ensemble de $\{0, 1\}^*$.

Une machine de Turing reçoit en entrée une chaîne $x \in A^*$ pour un alphabet A fixé. Cette entrée est complétée par des blancs (notés $\sqcup$) pour former une chaîne infinie dans les deux sens $s \in (A \cup \{\sqcup\})^{\mathbb{Z}}$. Cette chaîne s peut être considérée comme une bande avec une tête de lecture-écriture. À chaque étape, on peut lire et modifier seulement une composante à une certaine position, et la tête de lecture-écriture peut n'être déplacée que d'une seule position à chaque étape.

Une machine de Turing se compose d'un ensemble de $N + 1$ instructions numérotées $0, \ldots, N$. Au départ, l'instruction 0 est exécutée et la position actuelle de la chaîne est la position 1. Puis, chaque instruction correspond à la procédure suivante : lire le bit situé à la position actuelle et, suivant sa valeur, faire ce qui suit : remplacer le bit actuel par un élément de $A \cup \{\sqcup\}$, déplacer ou non la position actuelle d'une place vers la gauche ou vers la droite, et aller à l'instruction suivante.

Une instruction particulière, notée -1, représente l'arrêt du calcul. Les composantes de notre chaîne infinie s indexées par $1, 2, 3, \ldots$ jusqu'au premier $\sqcup$ correspondent alors à la chaîne obtenue en sortie de la machine. Formellement, on définit une machine de Turing de la manière suivante :

Définition 15.2. (Turing [1936]) *Soit* A *un alphabet et* $\bar{A} := A \cup \{\sqcup\}$*. Une* **machine de Turing** *(sur l'alphabet* A*) est définie par une fonction*

$$\Phi : \{0, \ldots, N\} \times \bar{A} \to \{-1, \ldots, N\} \times \bar{A} \times \{-1, 0, 1\}$$

pour un certain $N \in \mathbb{Z}_+$*. Le* **calcul** *de* Φ *sur l'entrée* x*, où* $x \in A^*$*, est la suite finie ou infinie de triplets* $(n^{(i)}, s^{(i)}, \pi^{(i)})$ *tels que* $n^{(i)} \in \{-1, \ldots, N\}$*,* $s^{(i)} \in \bar{A}^{\mathbb{Z}}$

et $\pi^{(i)} \in \mathbb{Z}$ ($i = 0, 1, 2, \ldots$) définie récursivement de la façon suivante ($n^{(i)}$ désigne l'instruction en cours, $s^{(i)}$ représente la chaîne, et $\pi^{(i)}$ est la position en cours) :

$n^{(0)} := 0$. $s_j^{(0)} := x_j$ *pour* $1 \leq j \leq \mathrm{taille}(x)$, *et* $s_j^{(0)} := \sqcup$ *pour tout* $j \leq 0$ *et* $j > \mathrm{taille}(x)$. $\pi^{(0)} := 1$.

Si $(n^{(i)}, s^{(i)}, \pi^{(i)})$ est déjà défini, on distingue deux cas. Si $n^{(i)} \neq -1$, alors soit $(m, \sigma, \delta) := \Phi\big(n^{(i)}, s^{(i)}_{\pi^{(i)}}\big)$ et posons $n^{(i+1)} := m$, $s^{(i+1)}_{\pi^{(i)}} := \sigma$, $s_j^{(i+1)} := s_j^{(i)}$ pour $j \in \mathbb{Z} \setminus \{\pi^{(i)}\}$, et $\pi^{(i+1)} := \pi^{(i)} + \delta$.

Si $n^{(i)} = -1$, alors la suite d'instructions est terminée. On définit alors $\mathrm{time}(\Phi, x) := i$ *et* $\mathrm{output}(\Phi, x) \in A^k$, *où* $k := \min\big\{j \in \mathbb{N} : s_j^{(i)} = \sqcup\big\} - 1$, *par* $\mathrm{output}(\Phi, x)_j := s_j^{(i)}$ *pour* $j = 1, \ldots, k$.

Si cette suite d'instructions est infinie (c.-à-d. $n^{(i)} \neq -1$ pour tout i), alors on pose $\mathrm{time}(\Phi, x) := \infty$. *Dans ce cas* $\mathrm{output}(\Phi, x)$ *n'est pas défini.*

Bien entendu, on s'intéresse surtout aux machines de Turing dont le calcul est fini ou même polynomialement borné :

Définition 15.3. *Soient A un alphabet, $S, T \subseteq A^*$ deux langages, et $f : S \to T$ une fonction. Soit Φ une machine de Turing avec alphabet A telle que* $\mathrm{time}(\Phi, s) < \infty$ *et* $\mathrm{output}(\Phi, s) = f(s)$ *pour chaque $s \in S$. On dit alors que Φ* **calcule** *f. S'il existe un polynôme p tel que, pour tout $s \in S$, on ait* $\mathrm{time}(\Phi, s) \leq p(\mathrm{taille}(s))$, *alors Φ est une* **machine de Turing polynomiale**.

Dans le cas où $S = A^*$ et $T = \{0, 1\}$, on dit que Φ **décide** *le langage $L := \{s \in S : f(s) = 1\}$. S'il existe une machine de Turing polynomiale calculant une fonction f (respectivement, décidant un langage L), alors on dit que f est* **calculable en temps polynomial** *(respectivement, L est* **décidable en temps polynomial***).*

Nous donnons maintenant un exemple afin de rendre ces définitions plus claires. La machine de Turing suivante $\Phi : \{0, \ldots, 4\} \times \{0, 1, \sqcup\} \to \{-1, \ldots, 4\} \times \{0, 1, \sqcup\} \times \{-1, 0, 1\}$ calcule la fonction successeur $f(n) = n + 1$ ($n \in \mathbb{N}$), où les nombres sont codés respectivement dans leur représentation binaire habituelle.

$\Phi(0,0)$	$=$	$(0,0,1)$	⓪	**While** $s_\pi \neq \sqcup$ **do** $\pi := \pi + 1$.
$\Phi(0,1)$	$=$	$(0,1,1)$		
$\Phi(0,\sqcup)$	$=$	$(1,\sqcup,-1)$		Poser $\pi := \pi - 1$.
$\Phi(1,1)$	$=$	$(1,0,-1)$	①	**While** $s_\pi = 1$ **do** $s_\pi := 0$ et $\pi := \pi - 1$.
$\Phi(1,0)$	$=$	$(-1,1,0)$		**If** $s_\pi = 0$ **then** $s_\pi := 1$ et **stop**.
$\Phi(1,\sqcup)$	$=$	$(2,\sqcup,1)$		Poser $\pi := \pi + 1$.
$\Phi(2,0)$	$=$	$(2,0,1)$	②	**While** $s_\pi = 0$ **do** $\pi := \pi + 1$.
$\Phi(2,\sqcup)$	$=$	$(3,0,-1)$		Poser $s_\pi := 0$ et $\pi := \pi - 1$.
$\Phi(3,0)$	$=$	$(3,0,-1)$	③	**While** $s_\pi = 0$ **do** $\pi := \pi - 1$.
$\Phi(3,\sqcup)$	$=$	$(4,\sqcup,1)$		Poser $\pi := \pi + 1$.
$\Phi(4,0)$	$=$	$(-1,1,0)$	④	Poser $s_\pi := 1$ et **stop**.

Notons que plusieurs valeurs de Φ ne sont pas précisées, car elles ne sont jamais utilisées quel que soit le calcul effectué. Les commentaires sur le côté droit détaillent les étapes du calcul. Les instructions ②, ③ et ④ sont utilisées seulement si l'entrée ne contient que des 1, c.-à-d. $n = 2^k - 1$ pour un $k \in \mathbb{Z}_+$. On a $\mathrm{time}(\Phi, s) \leq 4\,\mathrm{taille}(s) + 5$ pour toutes les entrées s, donc Φ est une machine de Turing polynomiale.

Au paragraphe suivant, nous montrerons que la définition précédente est compatible avec notre définition informelle d'un algorithme polynomial, donnée au paragraphe 1.2. Chaque algorithme polynomial de ce livre peut être simulé par une machine de Turing polynomiale.

15.2 Thèse de Church

La machine de Turing est le modèle théorique le plus usuel pour décrire les algorithmes. Bien qu'il puisse sembler très limité, il est aussi puissant que n'importe quel autre modèle raisonnable : l'ensemble des fonctions calculables (également appelées parfois **fonctions récursives**) est toujours le même. Cette affirmation, connue sous le nom de thèse de Church, est bien sûr trop imprécise pour être prouvée. Cependant, il existe des résultats puissants appuyant cette thèse. Par exemple, tout programme écrit dans un langage de programmation courant, comme le langage C, peut être modélisé par une machine de Turing. En particulier, tous les algorithmes de ce livre peuvent être réécrits en termes de machines de Turing. Cela est en général très incommode (aussi nous ne le ferons jamais), mais théoriquement possible. De plus, toute fonction calculable en temps polynomial avec un programme écrit en langage C est aussi calculable en temps polynomial avec une machine de Turing.

L'implémentation de programmes complexes sur une machine de Turing n'étant pas une tâche facile, nous considérons, comme étape intermédiaire, une machine de Turing avec deux bandes et deux têtes de lecture-écriture indépendantes (une pour chaque bande) :

Définition 15.4. *Soit A un alphabet et $\bar{A} := A \cup \{\sqcup\}$. Une* **machine de Turing à deux bandes** *est définie par une fonction*

$$\Phi : \{0, \ldots, N\} \times \bar{A}^2 \to \{-1, \ldots, N\} \times \bar{A}^2 \times \{-1, 0, 1\}^2$$

pour un certain $N \in \mathbb{Z}_+$. Le **calcul** *de Φ sur l'entrée x, où $x \in A^*$, est la suite finie ou infinie de quintuplets $(n^{(i)}, s^{(i)}, t^{(i)}, \pi^{(i)}, \rho^{(i)})$ tels que $n^{(i)} \in \{-1, \ldots, N\}$, $s^{(i)}, t^{(i)} \in \bar{A}^{\mathbb{Z}}$ et $\pi^{(i)}, \rho^{(i)} \in \mathbb{Z}$ ($i = 0, 1, 2, \ldots$) définie récursivement de la façon suivante :*

$n^{(0)} := 0$. $s_j^{(0)} := x_j$ pour $1 \leq j \leq \mathrm{taille}(x)$, et $s_j^{(0)} := \sqcup$ pour tout $j \leq 0$ et $j > \mathrm{taille}(x)$. $t_j^{(0)} := \sqcup$ pour tout $j \in \mathbb{Z}$. $\pi^{(0)} := 1$ et $\rho^{(0)} := 1$.

Si $(n^{(i)}, s^{(i)}, t^{(i)}, \pi^{(i)}, \rho^{(i)})$ est déjà défini, on distingue deux cas. Si $n^{(i)} \neq -1$, alors soit $(m, \sigma, \tau, \delta, \epsilon) := \Phi\left(n^{(i)}, s^{(i)}_{\pi^{(i)}}, t^{(i)}_{\rho^{(i)}}\right)$ et posons $n^{(i+1)} := m$, $s^{(i+1)}_{\pi^{(i)}} :=$

σ, $s_j^{(i+1)} := s_j^{(i)}$ pour $j \in \mathbb{Z}\setminus\{\pi^{(i)}\}$, $t_{\rho^{(i)}}^{(i+1)} := \tau$, $t_j^{(i+1)} := t_j^{(i)}$ pour $j \in \mathbb{Z}\setminus\{\rho^{(i)}\}$, $\pi^{(i+1)} := \pi^{(i)} + \delta$, et $\rho^{(i+1)} := \rho^{(i)} + \epsilon$.

Si $n^{(i)} = -1$, la suite d'instructions est terminée. $\mathrm{time}(\Phi, x)$ *et* $\mathrm{output}(\Phi, x)$ *sont définis de la même façon que pour une machine de Turing à une bande.*

Les machines de Turing avec plus de deux bandes se définissent de manière similaire, mais nous n'en aurons pas besoin. Avant de montrer comment nous réalisons des opérations standards avec une machine de Turing à deux bandes, nous prouvons d'abord qu'une machine de Turing à deux bandes peut être simulée par une machine de Turing ordinaire (à une bande).

Théorème 15.5. *Soit A un alphabet, et soit*

$$\Phi : \{0, \ldots, N\} \times (A \cup \{\sqcup\})^2 \to \{-1, \ldots, N\} \times (A \cup \{\sqcup\})^2 \times \{-1, 0, 1\}^2$$

une machine de Turing à deux bandes. Alors il existe un alphabet $B \supseteq A$ et une machine de Turing (à une bande)

$$\Phi' : \{0, \ldots, N'\} \times (B \cup \{\sqcup\}) \to \{-1, \ldots, N'\} \times (B \cup \{\sqcup\}) \times \{-1, 0, 1\}$$

telle que $\mathrm{output}(\Phi', x) = \mathrm{output}(\Phi, x)$ *et* $\mathrm{time}(\Phi', x) = O(\mathrm{time}(\Phi, x))^2$ *pour* $x \in A^*$.

Preuve. Nous utilisons les lettres s et t pour désigner les deux chaînes de Φ, et notons par π et ρ les positions des deux têtes de lecture-écriture, comme dans la définition 15.4. La chaîne de Φ' sera notée u et la position de sa tête de lecture-écriture par ψ.

Nous devons coder les deux chaînes s, t et les deux positions des têtes de lecture-écriture π, ρ à l'aide d'une seule chaîne u. Afin de rendre cela possible, chaque composante u_j de u est un quadruplet (s_j, p_j, t_j, r_j), où s_j et t_j sont les composantes correspondantes de s et t, et $p_j, r_j \in \{0, 1\}$ indiquent si les têtes de lecture-écriture de la première et de la seconde chaîne examinent actuellement la position j ; c.-à-d. que l'on a $p_j = 1$ si et seulement si $\pi = j$, et $r_j = 1$ si et seulement si $\rho = j$.

On définit donc $\bar{B} := (\bar{A} \times \{0, 1\} \times \bar{A} \times \{0, 1\})$; on identifie alors $a \in \bar{A}$ avec $(a, 0, \sqcup, 0)$ pour permettre de représenter des entrées de A^*. La première étape de Φ' consiste à initialiser les repères p_1 et r_1 à 1 :

$$\Phi'(0, (., 0, ., 0)) \;=\; (1, (., 1, ., 1)), 0) \qquad ⓪ \quad \text{Poser } \pi := \psi \text{ et } \rho := \psi.$$

Un point représente ici une valeur arbitraire (qui n'est pas modifiée).

Nous allons maintenant montrer comment implémenter une instruction générale $\Phi(m, \sigma, \tau) = (m', \sigma', \tau', \delta, \epsilon)$. Nous devons d'abord trouver les positions π et ρ. Il est plus simple de supposer que notre unique tête de lecture-écriture ψ se situe déjà à la position π ou ρ la plus à gauche ; c.-à-d. $\psi = \min\{\pi, \rho\}$. Nous devons alors trouver l'autre position en parcourant la chaîne u vers la droite. Puis nous devons vérifier si $s_\pi = \sigma$ et $t_\rho = \tau$ et, si c'est le cas, effectuer l'opération requise (écrire les nouvelles composantes de s et t, modifier π et ρ et passer à l'instruction suivante).

Le bloc suivant correspond à l'implémentation d'une instruction $\Phi(m,\sigma,\tau) = (m',\sigma',\tau',\delta,\epsilon)$ pour $m = 0$. Pour chaque m, on a $|\bar{A}|^2$ blocs de ce type (un par choix de σ et τ). Le second bloc pour $m = 0$ commence avec ⑬, le premier bloc pour m' avec Ⓜ, où $M := 12|\bar{A}|^2 m' + 1$. En tout, $N' = 12(N+1)|\bar{A}|^2$.

Un point représente encore une valeur arbitraire qui n'est pas modifiée. De même, ζ et ξ représentent un élément arbitraire de $\bar{A} \setminus \{\sigma\}$ et $\bar{A} \setminus \{\tau\}$, respectivement. Nous supposons que $\psi = \min\{\pi,\rho\}$ au début ; notons que ⑩, ⑪ et ⑫ garantissent que cette propriété est aussi vérifiée à la fin.

$\Phi'(1,(\zeta,1,.,.)) = (13,(\zeta,1,.,.),0)$	①	**If** $\psi = \pi$ et $s_\psi \neq \sigma$ **then go to** ⑬.
$\Phi'(1,(.,.,\xi,1)) = (13,(.,.,\xi,1),0)$		**If** $\psi = \rho$ et $t_\psi \neq \tau$ **then go to** ⑬.
$\Phi'(1,(\sigma,1,\tau,1)) = (2,(\sigma,1,\tau,1),0)$		**If** $\psi = \pi$ **then go to** ②.
$\Phi'(1,(\sigma,1,.,0)) = (2,(\sigma,1,.,0),0)$		
$\Phi'(1,(.,0,\tau,1)) = (6,(.,0,\tau,1),0)$		**If** $\psi = \rho$ **then go to** ⑥.
$\Phi'(2,(.,.,.,0)) = (2,(.,.,.,0),1)$	②	**While** $\psi \neq \rho$ **do** $\psi := \psi + 1$.
$\Phi'(2,(.,.,\xi,1)) = (12,(.,.,\xi,1),-1)$		**If** $t_\psi \neq \tau$ **then** poser $\psi := \psi - 1$ et **go to** ⑫.
$\Phi'(2,(.,.,\tau,1)) = (3,(.,.,\tau',0),\epsilon)$		Poser $t_\psi := \tau'$ et $\psi := \psi + \epsilon$.
$\Phi'(3,(.,.,.,0)) = (4,(.,.,.,1),1)$	③	Poser $\rho := \psi$ et $\psi := \psi + 1$.
$\Phi'(4,(.,0,.,.)) = (4,(.,0,.,.),-1)$	④	**While** $\psi \neq \pi$ **do** $\psi := \psi - 1$.
$\Phi'(4,(\sigma,1,.,.)) = (5,(\sigma',0,.,.),\delta)$		Poser $s_\psi := \sigma'$ et $\psi := \psi + \delta$.
$\Phi'(5,(.,0,.,.)) = (10,(.,1,.,.),-1)$	⑤	Poser $\pi := \psi$ et $\psi := \psi - 1$. **Go to** ⑩.
$\Phi'(6,(.,0,.,.)) = (6,(.,0,.,.),1)$	⑥	**While** $\psi \neq \pi$ **do** $\psi := \psi + 1$.
$\Phi'(6,(\zeta,1,.,.)) = (12,(\zeta,1,.,.),-1)$		**If** $s_\psi \neq \sigma$ **then** poser $\psi := \psi - 1$ et **go to** ⑫.
$\Phi'(6,(\sigma,1,.,.,)) = (7,(\sigma',0,.,.),\delta)$		Poser $s_\psi := \sigma'$ et $\psi := \psi + \delta$.
$\Phi'(7,(.,0,.,.)) = (8,(.,1,.,.),1)$	⑦	Poser $\pi := \psi$ et $\psi := \psi + 1$.
$\Phi'(8,(.,.,.,0)) = (8,(.,.,.,0),-1)$	⑧	**While** $\psi \neq \rho$ **do** $\psi := \psi - 1$.
$\Phi'(8,(.,.,\tau,1)) = (9,(.,.,\tau',0),\epsilon)$		Poser $t_\psi := \tau'$ et $\psi := \psi + \epsilon$.
$\Phi'(9,(.,.,.,0)) = (10,(.,.,.,1),-1)$	⑨	Poser $\rho := \psi$ et $\psi := \psi - 1$.
$\Phi'(10,(.,.,.,.)) = (11,(.,.,.,.),-1)$	⑩	Poser $\psi := \psi - 1$.
$\Phi'(11,(.,0,.,0)) = (11,(.,0,.,0),1)$	⑪	**While** $\psi \notin \{\pi,\rho\}$ **do** $\psi := \psi + 1$.
$\Phi'(11,(.,1,.,.)) = (M,(.,1,.,.),0)$		**Go to** Ⓜ.
$\Phi'(11,(.,0,.,1)) = (M,(.,0,.,1),0)$		
$\Phi'(12,(.,0,.,0)) = (12,(.,0,.,0),-1)$	⑫	**While** $\psi \notin \{\pi,\rho\}$ **do** $\psi := \psi - 1$.
$\Phi'(12,(.,1,.,.)) = (13,(.,1,.,.),0)$		
$\Phi'(12,(.,.,.,1)) = (13,(.,.,.,1),0)$		

Tout calcul de Φ' fait appel à au plus $|\bar{A}|^2$ blocs comme le précédent pour chaque étape de calcul de Φ. Il y a au plus $2|\pi - \rho| + 10$ étapes de calcul au sein de chaque bloc. Comme $|\bar{A}|$ est une constante et que $|\pi - \rho|$ est borné par $\mathrm{time}(\Phi, x)$, nous concluons que le calcul complet de Φ peut être simulé par Φ' en $O\left((\mathrm{time}(\Phi, x))^2\right)$ étapes.

Enfin, nous devons réarranger la chaîne obtenue en sortie, c.-à-d. remplacer chaque composante $(\sigma, ., ., .)$ par $(\sigma, 0, \sqcup, 0)$. Il est clair que cela peut au plus doubler le nombre d'étapes. □

Il n'est pas difficile d'implémenter des instructions plus complexes, et donc tout type d'algorithme, à l'aide d'une machine de Turing à deux bandes :

Nous utilisons l'alphabet $A = \{0, 1, \#\}$ et modélisons un nombre quelconque de variables par la chaîne

$$x_0\#\#1\#x_1\#\#10\#x_2\#\#11\#x_3\#\#100\#x_4\#\#101\#x_5\#\#\ldots \qquad (15.1)$$

que nous stockons sur la première bande. Chaque groupe contient une représentation binaire de l'indice i, suivie par la valeur de x_i, que nous supposons être une chaîne binaire. La première variable x_0 et la seconde bande sont utilisées uniquement comme des registres pour conserver les résultats intermédiaires des étapes de calcul.

Il n'est pas possible d'accéder aléatoirement aux variables, en temps constant, avec une machine de Turing, quel que soit le nombre de bandes dont nous disposons. Si nous simulons un algorithme arbitraire avec une machine de Turing à deux bandes, nous devrons examiner la première bande assez souvent. De plus, si la longueur de la chaîne change pour une variable, la sous-chaîne située à droite de cette variable doit être déplacée. Cependant, chaque opération standard (c.-à-d. chaque étape élémentaire d'un algorithme) peut être simulée à l'aide d'une machine de Turing à deux bandes et nécessite alors $O(l^2)$ étapes de calcul, où l est la longueur actuelle de la chaîne (15.1).

Nous allons essayer de rendre cela plus clair à l'aide d'un exemple concret. Considérons l'instruction suivante : ajouter à x_5 la valeur de la variable dont l'indice est donné par x_2.

Pour obtenir la valeur de x_5, nous recherchons d'abord sur la première bande la sous-chaîne $\#\#101\#$. Nous considérons alors les éléments situés à droite de $\#\#101\#$ jusqu'au symbole $\#$ suivant. Ils constituent une sous-chaîne que nous copions sur la seconde bande. Cela est facile, car nous avons deux têtes de lecture-écriture distinctes. Nous copions alors la chaîne de la seconde bande en x_0. Si la nouvelle valeur de x_0 est plus courte ou plus longue que la précédente, nous devons déplacer de manière appropriée le reste de la chaîne (15.1) vers la gauche ou vers la droite.

Ensuite, nous devons chercher l'indice de la variable, donné par x_2. Pour ce faire, nous copions d'abord x_2 sur la seconde bande. Nous examinons alors la première bande et contrôlons l'indice de chaque variable (on le compare bit à bit à la chaîne de la seconde bande). Une fois que nous avons trouvé le bon indice, nous copions la valeur de la variable correspondante sur la seconde bande.

Ajoutons maintenant le nombre conservé en x_0 à celui qui est sur la seconde bande. Il n'est pas difficile de concevoir une machine de Turing pour effectuer cette tâche, en utilisant la méthode standard. Nous pouvons remplacer le nombre sur la seconde bande par le résultat que nous obtenons lors du calcul. Finalement, nous avons le résultat sur la seconde bande et le recopions en x_5. Nous déplaçons, si besoin est, de manière appropriée la sous-chaîne située à droite de x_5.

Toutes les opérations précédentes peuvent être effectuées à l'aide d'une machine de Turing à deux bandes en $O(l^2)$ étapes de calcul (en fait, toutes les opérations, à part les déplacements de la chaîne (15.1), peuvent se faire en $O(l)$ étapes). Il devrait être clair que c'est également le cas pour toutes les opérations standards, multiplication et division comprises.

D'après la définition 1.4, on dit qu'un algorithme s'exécute en temps polynomial s'il existe $k \in \mathbb{N}$ tel que le nombre d'étapes élémentaires soit borné par $O(n^k)$ et que tout nombre, utilisé dans un calcul intermédiaire, puisse être stocké à l'aide de $O(n^k)$ bits, où n est la taille de l'entrée. De plus, nous stockons au plus $O(n^k)$ nombres au cours du calcul. Ainsi, nous pouvons borner la longueur de chacune des deux chaînes d'une machine de Turing à deux bandes, simulant un tel algorithme, par $l = O(n^k \cdot n^k) = O(n^{2k})$, et ainsi borner le temps de calcul par $O(n^k(n^{2k})^2) = O(n^{5k})$. Cela est encore polynomial par rapport à la taille de l'entrée.

On peut alors conclure, en utilisant le théorème 15.5, que pour toute fonction chaîne f, il existe un algorithme polynomial calculant f si et seulement s'il existe une machine de Turing polynomiale calculant f.

Hopcroft et Ullman [1979], Lewis et Papadimitriou [1981] et Van Emde Boas [1990] fournissent plus de détails sur l'équivalence entre différents modèles de machines. Un autre modèle courant (qui est proche du modèle informel que nous avons donné au paragraphe 1.2) est la machine RAM (voir exercice 3) qui permet de réaliser des opérations arithmétiques sur les entiers en temps constant. D'autres modèles n'autorisent que les opérations sur les bits (ou des entiers de longueur fixée), ce qui est plus réaliste lorsque l'on traite de grands nombres. Évidemment, l'addition et la comparaison de nombres entiers codés par des mots de n bits peuvent être réalisées à l'aide de $O(n)$ opérations sur les bits. Pour la multiplication (et la division), la méthode la plus évidente requiert $O(n^2)$opérations, mais l'algorithme de Schönhage et Strassen [1971] nécessite seulement $O(n \log n \log\log n)$ opérations sur les bits pour multiplier deux entiers codés par des mots de n bits, et cela a été encore amélioré par Fürer [2007]. Cela implique bien sûr l'existence d'algorithmes pour l'addition et la comparaison de nombres rationnels avec la même complexité. Dans le domaine de la calculabilité en temps polynomial, tous les modèles sont équivalents, mais, bien entendu, les temps de calcul varient beaucoup d'un modèle à un autre.

En principe, la méthode consistant à représenter les entrées par des chaînes en 0-1 (ou des chaînes sur un certain alphabet fixé) permet de travailler avec tout type de nombres réels, par exemple les nombres algébriques (si $x \in \mathbb{R}$ est la k-ième plus petite racine d'un polynôme p, alors x peut être représenté par une liste contenant k et les degrés et les coefficients de p). Cependant, il n'est pas possible de

représenter des nombres réels quelconques dans un calculateur numérique, car il n'y a qu'un nombre dénombrable de chaînes en 0-1 alors que les nombres réels sont indénombrables. Nous suivons l'approche classique et nous nous limitons à des entrées correspondant à des nombres rationnels dans ce chapitre.

Nous terminons ce paragraphe en donnant une définition formelle des algorithmes fondés sur un oracle, reposant sur les machines de Turing à deux bandes. Nous pouvons faire appel à un oracle à n'importe quel stade du calcul ; nous utilisons la seconde bande pour écrire les entrées de l'oracle et lire ses sorties. Nous introduisons une instruction particulière notée -2 pour représenter les appels à l'oracle :

Définition 15.6. *Soit A un alphabet et $\bar{A} := A \cup \{\sqcup\}$. Soit $X \subseteq A^*$, et soit $f(x) \subseteq A^*$ un langage non vide pour chaque $x \in X$. Une* **machine de Turing fondée sur un oracle** *utilisant f est une application*

$$\Phi : \{0, \ldots, N\} \times \bar{A}^2 \to \{-2, \ldots, N\} \times \bar{A}^2 \times \{-1, 0, 1\}^2$$

pour un certain $N \in \mathbb{Z}_+$; son calcul est défini de la même façon que pour une machine de Turing à deux bandes, mais avec une différence : si, pour une étape de calcul i, $\Phi\left(n^{(i)}, s^{(i)}_{\pi^{(i)}}, t^{(i)}_{\rho^{(i)}}\right) = (-2, \sigma, \tau, \delta, \epsilon)$ pour un quadruplet $\sigma, \tau, \delta, \epsilon$, alors soit la chaîne sur la seconde bande $x \in A^k$, où $k := \min\left\{j \in \mathbb{N} : t^{(i)}_j = \sqcup\right\} - 1$, qui est définie par $x_j := t^{(i)}_j$ pour $j = 1, \ldots, k$. Si $x \in X$, alors la seconde bande est remplacée par $t^{(i+1)}_j = y_j$ pour $j = 1, \ldots, \text{taille}(y)$ et $t^{(i+1)}_{\text{taille}(y)+1} = \sqcup$ pour un certain $y \in f(x)$. Le reste n'est pas modifié, et le calcul continue avec $n^{(i+1)} := n^{(i)} + 1$ (et s'arrête si $n^{(i)} = -1$).

Toutes les définitions qui concernent les machines de Turing peuvent être étendues aux machines de Turing fondées sur un oracle. La réponse fournie par un oracle n'est pas nécessairement unique ; ainsi, il peut y avoir plusieurs calculs possibles pour une même entrée. Lorsque nous prouvons l'exactitude d'un algorithme fondé sur un oracle ou que nous voulons estimer son temps de calcul, nous devons tenir compte de tous les calculs possibles, c.-à-d. de tous les choix possibles de l'oracle.

D'après les résultats de ce paragraphe, l'existence d'un algorithme polynomial (fondé sur un oracle) est équivalente à l'existence d'une machine de Turing polynomiale (fondée sur un oracle).

15.3 *P* et *NP*

La théorie de la complexité se concentre essentiellement sur les problèmes de décision. Tout langage $L \subseteq \{0, 1\}^*$ peut être lié à un problème de décision : étant donné une chaîne en 0-1, décider si elle appartient à L. Cependant, nous nous intéressons davantage à des problèmes du type suivant :

PROBLÈME DU CYCLE HAMILTONIEN	
Instance	Un graphe non orienté G.
Question	G possède-t-il un cycle hamiltonien ?

Nous supposerons toujours qu'un codage efficace de l'entrée, sous la forme d'une chaîne binaire, est fixé. De temps en temps, nous étendrons notre alphabet avec d'autres symboles. Par exemple, nous supposons qu'un graphe est décrit par une liste d'adjacence, et qu'une telle liste peut être codée facilement par une chaîne binaire de longueur $O(n+m\log n)$, où n et m représentent les nombres de sommets et d'arêtes du graphe. Nous supposons toujours l'existence d'un codage efficace, c.-à-d. d'un codage dont la longueur est polynomialement bornée par la longueur minimale possible du codage.

Les chaînes binaires ne correspondent pas toutes à des instances du PROBLÈME DU CYCLE HAMILTONIEN. Seules les chaînes binaires représentant un graphe non orienté sont des instances possibles. Pour la plupart des problèmes de décision intéressants, les instances correspondent à un sous-ensemble propre des chaînes en 0-1. Nous avons besoin de supposer que l'on peut décider en temps polynomial si une chaîne arbitraire correspond à une instance ou non :

Définition 15.7. *Un* **problème de décision** *est une paire $\mathcal{P} = (X, Y)$, où X est un langage décidable en temps polynomial et $Y \subseteq X$. Les éléments de X sont appelés des* **instances** *de $\mathcal{P}$; les éléments de Y sont des* **instances-«oui»**, *ceux de $X \setminus Y$ sont des* **instances-«non»**.

Un algorithme pour un problème de décision (X, Y) est un algorithme qui calcule la fonction $f : X \to \{0, 1\}$, définie par $f(x) = 1$ pour $x \in Y$ et $f(x) = 0$ pour $x \in X \setminus Y$.

Nous donnons deux exemples supplémentaires, les problèmes de décision correspondant à la PROGRAMMATION LINÉAIRE et à la PROGRAMMATION EN NOMBRES ENTIERS :

PROBLÈME DES INÉGALITÉS LINÉAIRES	
Instance	Une matrice $A \in \mathbb{Z}^{m\times n}$ et un vecteur $b \in \mathbb{Z}^m$.
Question	Existe-t-il un vecteur $x \in \mathbb{Q}^n$ tel que $Ax \leq b$?

PROBLÈME DES INÉGALITÉS LINÉAIRES EN NOMBRES ENTIERS	
Instance	Une matrice $A \in \mathbb{Z}^{m\times n}$ et un vecteur $b \in \mathbb{Z}^m$.
Question	Existe-t-il un vecteur $x \in \mathbb{Z}^n$ tel que $Ax \leq b$?

Définition 15.8. *On note P la classe de tous les problèmes de décision pour lesquels il existe un algorithme polynomial.*

Autrement dit, un membre de *P* est un couple (X, Y) tel que $Y \subseteq X \subseteq \{0, 1\}^*$ où X et Y sont tous deux des langages décidables en temps polynomial. En général

pour prouver qu'un problème est dans *P*, on décrit un algorithme polynomial résolvant ce problème. D'après les résultats du paragraphe 15.2, il existe une machine de Turing polynomiale pour tout problème dans *P*. D'après le théorème de Khachiyan 4.18, le problème de décision associé aux INÉGALITÉS LINÉAIRES appartient à *P*. En revanche, on ne sait pas si les problèmes de décision associés aux INÉGALITÉS LINÉAIRES EN NOMBRES ENTIERS ou au PROBLÈME DU CYCLE HAMILTONIEN appartiennent à *P*. Nous allons maintenant introduire une autre classe, notée *NP*, qui contient ces deux problèmes, et en fait la plupart des problèmes de décision présentés dans ce livre.

Pour les problèmes de la classe NP, nous n'exigeons pas un algorithme polynomial, en revanche nous demandons qu'il y ait, pour chaque instance-«oui», un certificat qui puisse être vérifié en temps polynomial. Par exemple, pour le PROBLÈME DU CYCLE HAMILTONIEN, la donnée d'un cycle hamiltonien constitue un certificat. Il est facile de vérifier si une chaîne donnée correspond à la représentation binaire d'un cycle hamiltonien. Notons que nous ne demandons pas de certificat pour les instances-«non». Nous définissons maintenant cette notion de manière formelle :

Définition 15.9. *Un problème de décision $\mathcal{P} = (X, Y)$ appartient à NP s'il existe un polynôme p et un problème de décision $\mathcal{P}' = (X', Y')$ dans P, où*

$$X' := \left\{ x\#c : x \in X,\, c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor} \right\},$$

tels que

$$Y = \left\{ y \in X : \textit{Il existe une chaîne } c \in \{0,1\}^{\lfloor p(\text{taille}(y))\rfloor} \textit{ telle que } y\#c \in Y' \right\}.$$

Ici $x\#c$ désigne la chaîne obtenue par concaténation de la chaîne x, du symbole $\#$ et de la chaîne c. Une chaîne c telle que $y\#c \in Y'$ est appelée un **certificat** *pour y (puisque c prouve que $y \in Y$). Un algorithme pour $\mathcal{P}'$ est appelé un* **algorithme de vérification de certificat**.

Proposition 15.10. *$P \subseteq NP$.*

Preuve. On peut choisir p identiquement nul. Un algorithme pour $\mathcal{P}'$ supprime simplement le dernier symbole de l'entrée «$x\#$» et applique ensuite un algorithme pour $\mathcal{P}$. □

On ne sait pas si $P = NP$. En fait, il s'agit du problème ouvert le plus important de la théorie de la complexité. Les problèmes suivants sont dans *NP*, mais ne sont pas connus pour être dans P :

Proposition 15.11. *Le* PROBLÈME DU CYCLE HAMILTONIEN *appartient à NP.*

Preuve. Pour chaque instance-«oui» G, tout cycle hamiltonien de G est un certificat. Il est évidemment possible de vérifier en temps polynomial si un ensemble d'arêtes donné forme un cycle hamiltonien. □

Proposition 15.12. *Le* PROBLÈME DES INÉGALITÉS LINÉAIRES EN NOMBRES ENTIERS *appartient à NP.*

Preuve. On prend simplement un vecteur solution comme certificat. S'il existe une solution, il en existe une de taille polynomiale d'après le corollaire 5.7. □

La notation *NP* signifie «non déterministe polynomial». Pour expliquer cela, nous devons définir ce qu'est un algorithme non déterministe. Nous allons en profiter pour définir plus généralement les algorithmes randomisés, concept que nous avons déjà mentionné précédemment. La caractéristique commune des algorithmes randomisés est que leur calcul ne dépend pas seulement de l'input, mais également de bits aléatoires.

Définition 15.13. *Un* **algorithme randomisé** *pour calculer une fonction $f : S \to T$ peut être défini comme un algorithme calculant une fonction $g : \{s\#r : s \in S, r \in \{0,1\}^{k(s)}\} \to T$. Ainsi, pour chaque instance $s \in S$, l'algorithme utilise $k(s) \in \mathbb{Z}_+$ bits aléatoires. Nous mesurons la dépendance entre le temps de calcul et $\text{taille}(s)$ uniquement ; les algorithmes randomisés, qui s'exécutent en temps polynomial, peuvent lire seulement un nombre polynomial de bits aléatoires.*

Naturellement, on s'intéresse à un tel algorithme randomisé seulement si f et g sont liées. Dans le cas idéal, où $g(s\#r) = f(s)$ pour tout $s \in S$ et tout $r \in \{0,1\}^{k(s)}$, on parle alors d'un **algorithme de Las Vegas***. Un algorithme de Las Vegas calcule toujours le résultat correct, seul le temps de calcul peut varier. Parfois des algorithmes encore moins déterministes peuvent être intéressants : s'il existe une probabilité strictement positive p d'avoir une réponse correcte, qui soit indépendante de l'instance, c.-à-d.*

$$p := \inf_{s \in S} \frac{|\{r \in \{0,1\}^{k(s)} : g(s\#r) = f(s)\}|}{2^{k(s)}} > 0,$$

on a alors un **algorithme de Monte-Carlo***.*

Si $T = \{0,1\}$, et que pour tout $s \in S$, tel que $f(s) = 0$, on a $g(s\#r) = 0$ pour tout $r \in \{0,1\}^{k(s)}$, alors on a un algorithme randomisé avec **erreur unilatérale***. Si de plus, pour tout $s \in S$, tel que $f(s) = 1$, il existe au moins un $r \in \{0,1\}^{k(s)}$ tel que $g(s\#r) = 1$, alors l'algorithme est appelé un* **algorithme non déterministe***.*

On peut également considérer qu'un algorithme randomisé est un algorithme fondé sur un oracle où l'oracle fournit un bit aléatoire (0 ou 1) à chaque appel. Un algorithme non déterministe, associé à un problème de décision, répond toujours «non» pour une instance-«non», et pour chaque instance-«oui», il y a une chance qu'il réponde «oui». Il est facile de faire l'observation suivante :

Proposition 15.14. *Un problème de décision appartient à NP si et seulement s'il existe un algorithme non déterministe polynomial le résolvant.*

Preuve. Soit $\mathcal{P} = (X, Y)$ un problème de décision dans *NP*, et soit $\mathcal{P}' = (X', Y')$ défini comme à la définition 15.9. Alors un algorithme polynomial pour $\mathcal{P}'$ est aussi

un algorithme non déterministe pour $\mathcal{P}$: le certificat inconnu est simplement remplacé par des bits aléatoires. Puisque le nombre de bits aléatoires est borné par un polynôme par rapport à $\text{taille}(x)$, $x \in X$, le temps de calcul de l'algorithme l'est également.

Inversement, s'il existe un algorithme non déterministe polynomial pour $\mathcal{P} = (X, Y)$, utilisant $k(x)$ bits aléatoires pour une instance x, alors il existe un polynôme p tel que $k(x) \leq p(\text{taille}(x))$ pour chaque instance x. On définit

$$X' := \left\{ x\#c : x \in X,\, c \in \{0,1\}^{\lfloor p(\text{taille}(x)) \rfloor} \right\}$$

et

$$Y' := \{ x\#c \in X' : g(x\#r) = 1,\ r \text{ est constitué des } k(x) \text{ premiers bits de } c \}.$$

Alors, d'après la définition des algorithmes non déterministes, on a $(X', Y') \in P$ et

$$Y = \left\{ y \in X : \text{Il existe une chaîne } c \in \{0,1\}^{\lfloor p(\text{taille}(y)) \rfloor} \text{ telle que } y\#c \in Y' \right\}.$$

□

La plupart des problèmes de décision rencontrés en optimisation combinatoire appartiennent à *NP*. Pour beaucoup d'entre eux, on ne sait pas s'il existe un algorithme polynomial. Cependant, on peut dire que certains problèmes sont au moins aussi difficiles que d'autres. Pour rendre cette notion plus précise, nous introduisons le concept important de réduction polynomiale.

Définition 15.15. *Soient $\mathcal{P}_1$ et $\mathcal{P}_2 = (X, Y)$ des problèmes de décision. Soit une fonction $f : X \to \{0,1\}$ telle que $f(x) = 1$ pour $x \in Y$ et $f(x) = 0$ pour $x \in X \setminus Y$. On dit que $\mathcal{P}_1$* **se réduit polynomialement** *à $\mathcal{P}_2$ s'il existe pour $\mathcal{P}_1$ un algorithme fondé sur un oracle polynomial utilisant f.*

L'importance du concept de réduction polynomiale est principalement justifiée par l'observation suivante :

Proposition 15.16. *Si $\mathcal{P}_1$ se réduit polynomialement à $\mathcal{P}_2$ et s'il existe un algorithme polynomial pour $\mathcal{P}_2$, alors il existe un algorithme polynomial pour $\mathcal{P}_1$.*

Preuve. Soit A_2 un algorithme pour $\mathcal{P}_2$ tel que $\text{time}(A_2, y) \leq p_2(\text{taille}(y))$ pour toute instance y de $\mathcal{P}_2$, et considérons $f(x) := \text{output}(A_2, x)$. Soit A_1 un algorithme fondé sur un oracle pour $\mathcal{P}_1$ utilisant f, tel que $\text{time}(A_1, x) \leq p_1(\text{taille}(x))$ pour toute instance x de $\mathcal{P}_1$. Alors, en remplaçant les appels à l'oracle dans A_1 par des sous-programmes équivalents à A_2, on obtient un algorithme A_3 pour $\mathcal{P}_1$. Pour toute instance x de $\mathcal{P}_1$ telle que $\text{taille}(x) = n$, on a $\text{time}(A_3, x) \leq p_1(n) \cdot p_2(p_1(n))$. En effet, il peut y avoir au plus $p_1(n)$ appels à l'oracle dans A_1, et aucune des instances de $\mathcal{P}_2$ produites par A_1 ne peut être plus longue que

$p_1(n)$. Puisque l'on peut choisir que p_1 et p_2 soient des polynômes, on en conclut que l'algorithme A_3 est polynomial. □

La théorie de la *NP*-complétude est fondée sur un type particulier de réduction polynomiale :

Définition 15.17. *Soient $\mathcal{P}_1 = (X_1, Y_1)$ et $\mathcal{P}_2 = (X_2, Y_2)$ des problèmes de décision. On dit que $\mathcal{P}_1$ se* **transforme polynomialement** *en $\mathcal{P}_2$, s'il existe une fonction $f : X_1 \to X_2$ calculable en temps polynomial, telle que $f(x_1) \in Y_2$ pour tout $x_1 \in Y_1$ et $f(x_1) \in X_2 \setminus Y_2$ pour tout $x_1 \in X_1 \setminus Y_1$.*

Autrement dit, les instances-«oui» sont transformées en instances-«oui», et les instances-«non» sont transformées en instances-«non». Évidemment, si un problème $\mathcal{P}_1$ se transforme polynomialement en $\mathcal{P}_2$, alors $\mathcal{P}_1$ se réduit aussi polynomialement à $\mathcal{P}_2$. Les transformations polynomiales sont quelquefois appelées des réductions de Karp, alors que les réductions polynomiales générales sont aussi connues sous le nom de réductions de Turing. Il est facile de voir que ces réductions sont toutes deux transitives.

Définition 15.18. *Un problème de décision $\mathcal{P} \in NP$ est dit* ***NP*-complet** *si tous les autres problèmes de NP se transforment polynomialement en $\mathcal{P}$.*

D'après la proposition 15.16, on sait que s'il existe un algorithme polynomial pour un problème *NP*-complet, alors $P = NP$.

Bien entendu, la définition précédente n'aurait pas de sens s'il n'existait pas de problème *NP*-complet. Le paragraphe suivant donne une preuve de l'existence d'un problème *NP*-complet.

15.4 Théorème de Cook

Dans ses travaux précurseurs, Cook [1971] a prouvé qu'un certain problème de décision, appelé SATISFAISABILITÉ, est en fait *NP*-complet. Nous avons besoin de quelques définitions :

Définition 15.19. *Supposons que $X = \{x_1, \ldots, x_k\}$ soit un ensemble de* **variables booléennes***. Un* **assignement** *pour X est une fonction $T : X \to \{vrai, faux\}$. On étend T à l'ensemble $L := X \dot\cup \{\overline{x} : x \in X\}$, en posant $T(\overline{x}) := vrai$ si $T(x) := faux$ et vice versa ($\overline{x}$ peut être considéré comme la négation de x). Les éléments de L sont appelés les* **littéraux** *sur X.*

Une **clause** *sur X est un ensemble de littéraux sur X. Une clause représente la disjonction de ces littéraux et est* **satisfaite** *par un assignement si et seulement si au moins un de ses membres est vrai. Une famille $\mathcal{Z}$ de clauses sur X est* **satisfaisable** *si et seulement s'il existe un assignement satisfaisant simultanément toutes ses clauses.*

Puisque nous considérons la conjonction de disjonctions de littéraux, nous parlons aussi de formules booléennes sous forme normale conjonctive. Par exemple, la famille $\{\{x_1, \overline{x_2}\}, \{\overline{x_2}, \overline{x_3}\}, \{x_1, x_2, \overline{x_3}\}, \{\overline{x_1}, x_3\}\}$ correspond à la formule booléenne $(x_1 \vee \overline{x_2}) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$. L'assignement $T(x_1) :=$ *vrai*, $T(x_2) :=$ *faux* et $T(x_3) :=$ *vrai* montre qu'elle est satisfaisable. Nous pouvons maintenant spécifier le PROBLÈME DE LA SATISFAISABILITÉ :

PROBLÈME DE LA SATISFAISABILITÉ	
Instance	Un ensemble X de variables et une famille $\mathcal{Z}$ de clauses sur X.
Question	$\mathcal{Z}$ est-elle satisfaisable ?

Théorème 15.20. (Cook [1971]) *Le* PROBLÈME DE LA SATISFAISABILITÉ *est NP-complet.*

Preuve. Le PROBLÈME DE LA SATISFAISABILITÉ appartient à *NP* car un assignement satisfaisant peut servir de certificat pour toute instance-«oui». Ce certificat peut bien entendu être vérifié en temps polynomial.

Soit maintenant $\mathcal{P} = (X, Y)$ un autre problème dans *NP*. Nous devons montrer que $\mathcal{P}$ se transforme polynomialement en PROBLÈME DE LA SATISFAISABILITÉ.

D'après la définition 15.9, il existe un polynôme p et un problème de décision $\mathcal{P}' = (X', Y')$ dans *P*, tels que $X' := \left\{x\#c : x \in X,\, c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor}\right\}$ et

$$Y \;=\; \left\{y \in X : \text{Il existe une chaîne } c \in \{0,1\}^{\lfloor p(\text{taille}(y))\rfloor} \text{ telle que } y\#c \in Y'\right\}.$$

Soit

$$\Phi : \{0, \ldots, N\} \times \bar{A} \;\to\; \{-1, \ldots, N\} \times \bar{A} \times \{-1, 0, 1\}$$

une machine de Turing polynomiale pour $\mathcal{P}'$ avec l'alphabet A. Considérons $\bar{A} := A \cup \{\sqcup\}$. Soit q un polynôme tel que $\text{time}(\Phi, x\#c) \leq q(\text{taille}(x\#c))$ pour toute instance $x\#c \in X'$. Notons que $\text{taille}(x\#c) = \text{taille}(x) + 1 + \lfloor p(\text{taille}(x))\rfloor$.

Nous allons maintenant construire, pour chaque $x \in X$, un ensemble $\mathcal{Z}(x)$ de clauses sur un certain ensemble $V(x)$ de variables booléennes, tel que $\mathcal{Z}(x)$ soit satisfaisable si et seulement si $x \in Y$.

Nous notons $Q := q(\text{taille}(x) + 1 + \lfloor p(\text{taille}(x))\rfloor)$. Q est une borne supérieure de la longueur de tout calcul de Φ sur l'input $x\#c$, pour tout $c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor}$. $V(x)$ contient les variables booléennes suivantes :

- une variable $v_{ij\sigma}$ pour tout $0 \leq i \leq Q$, $-Q \leq j \leq Q$ et $\sigma \in \bar{A}$;
- une variable w_{ijn} pour tout $0 \leq i \leq Q$, $-Q \leq j \leq Q$ et $-1 \leq n \leq N$.

Ces notations doivent être interprétées de la manière suivante : $v_{ij\sigma}$ indique si, à l'instant i (c.-à-d. après i étapes de calcul), la j-ième position de la chaîne contient le symbole σ, tandis que w_{ijn} indique si, à l'instant i, la j-ième position de la chaîne est examinée et la n-ième instruction est exécutée.

Ainsi, si $(n^{(i)}, s^{(i)}, \pi^{(i)})_{i=0,1,\ldots}$ correspond à un calcul de Φ, alors nous allons mettre $v_{ij\sigma}$ à *vrai* si et seulement si $s_j^{(i)} = \sigma$ et w_{ijn} à *vrai* si et seulement si $\pi^{(i)} = j$ et $n^{(i)} = n$.

L'ensemble $\mathcal{Z}(x)$ des clauses, que nous allons construire, sera satisfaisable si et seulement s'il existe une chaîne c telle que $\mathrm{output}(\Phi, x\#c) = 1$.

$\mathcal{Z}(x)$ contient les clauses suivantes qui permettent de modéliser les conditions suivantes :

À tout instant, chaque position de la chaîne contient un seul symbole :

- $\{v_{ij\sigma} : \sigma \in \bar{A}\}$ pour $0 \leq i \leq Q$ et $-Q \leq j \leq Q$;
- $\{\overline{v_{ij\sigma}}, \overline{v_{ij\tau}}\}$ pour $0 \leq i \leq Q$, $-Q \leq j \leq Q$ et $\sigma, \tau \in \bar{A}$ tels que $\sigma \neq \tau$.

À tout instant, une seule position de la chaîne est examinée et une seule instruction est exécutée :

- $\{w_{ijn} : -Q \leq j \leq Q, -1 \leq n \leq N\}$ pour $0 \leq i \leq Q$;
- $\{\overline{w_{ijn}}, \overline{w_{ij'n'}}\}$ pour $0 \leq i \leq Q$, $-Q \leq j, j' \leq Q$ et $-1 \leq n, n' \leq N$ tels que $(j, n) \neq (j', n')$.

L'algorithme démarre correctement avec l'input $x\#c$ pour un certain $c \in \{0,1\}^{\lfloor p(\mathrm{taille}(x))\rfloor}$:

- $\{v_{0,j,x_j}\}$ pour $1 \leq j \leq \mathrm{taille}(x)$;
- $\{v_{0,\mathrm{taille}(x)+1,\#}\}$;
- $\{v_{0,\mathrm{taille}(x)+1+j,0}, v_{0,\mathrm{taille}(x)+1+j,1}\}$ pour $1 \leq j \leq \lfloor p(\mathrm{taille}(x))\rfloor$;
- $\{v_{0,j,\sqcup}\}$ pour $-Q \leq j \leq 0$ et $\mathrm{taille}(x) + 2 + \lfloor p(\mathrm{taille}(x))\rfloor \leq j \leq Q$;
- $\{w_{010}\}$.

L'algorithme répond correctement :

- $\{\overline{v_{ij\sigma}}, \overline{w_{ijn}}, v_{i+1,j,\tau}\}$, $\{\overline{v_{ij\sigma}}, \overline{w_{ijn}}, w_{i+1,j+\delta,m}\}$ pour $0 \leq i < Q$, $-Q \leq j \leq Q$, $\sigma \in \bar{A}$ et $0 \leq n \leq N$, où $\Phi(n, \sigma) = (m, \tau, \delta)$.

Lorsque l'algorithme atteint l'instruction -1, il s'arrête :

- $\{\overline{w_{i,j,-1}}, w_{i+1,j,-1}\}$, $\{\overline{w_{i,j,-1}}, \overline{v_{i,j,\sigma}}, v_{i+1,j,\sigma}\}$ pour $0 \leq i < Q$, $-Q \leq j \leq Q$ et $\sigma \in \bar{A}$.

Les positions qui ne sont pas examinées restent inchangées :

- $\{\overline{v_{ij\sigma}}, \overline{w_{ij'n}}, v_{i+1,j,\sigma}\}$ pour $0 \leq i \leq Q$, $\sigma \in \bar{A}$, $-1 \leq n \leq N$ et $-Q \leq j, j' \leq Q$ tels que $j \neq j'$.

L'output de l'algorithme est 1 :

- $\{v_{Q,1,1}\}, \{v_{Q,2,\sqcup}\}$.

La longueur du codage de $\mathcal{Z}(x)$ est $O(Q^3 \log Q)$: il y a $O(Q^3)$ occurrences de littéraux, dont les indices nécessitent un espace $O(\log Q)$. Puisque Q dépend polynomialement de $\mathrm{taille}(x)$, nous pouvons en conclure qu'il existe un algorithme polynomial qui, étant donné x, construit $\mathcal{Z}(x)$. Notons que p, Φ et q sont fixés et ne font pas partie de l'input de cet algorithme.

Il reste à montrer que $\mathcal{Z}(x)$ est satisfaisable si et seulement si $x \in Y$.

Si $\mathcal{Z}(x)$ est satisfaisable, considérons un assignement T satisfaisant toutes les clauses. Soit $c \in \{0,1\}^{\lfloor p(\mathrm{taille}(x))\rfloor}$ tel que $c_j = 1$ pour tout j tel que

$T(v_{0,\text{taille}(x)+1+j,1}) = \textit{vrai}$ et $c_j = 0$ sinon. Par la construction précédente, les variables reflètent le calcul de Φ sur l'input $x\#c$. Ainsi nous pouvons conclure que $\text{output}(\Phi, x\#c) = 1$. Comme Φ est un algorithme de vérification de certificat, cela implique que x est une instance-«oui».

Inversement, si $x \in Y$, considérons un certificat c pour x.
Soit $(n^{(i)}, s^{(i)}, \pi^{(i)})_{i=0,1,\ldots,m}$ le calcul de Φ sur l'input $x\#c$. On définit alors $T(v_{i,j,\sigma}) := \textit{vrai}$ si et seulement si $s_j^{(i)} = \sigma$ et $T(w_{i,j,n}) = \textit{vrai}$ si et seulement si $\pi^{(i)} = j$ et $n^{(i)} = n$. Pour $i := m+1, \ldots, Q$, on pose $T(v_{i,j,\sigma}) := T(v_{i-1,j,\sigma})$ et $T(w_{i,j,n}) := T(w_{i-1,j,n})$ pour tout j, n et σ. Alors T est un assignement satisfaisant $\mathcal{Z}(x)$, ce qui complète la preuve. □

Le PROBLÈME DE LA SATISFAISABILITÉ n'est pas le seul problème *NP*-complet ; nous en rencontrerons beaucoup d'autres dans ce livre. Comme nous disposons maintenant d'un premier problème *NP*-complet, nous pouvons prouver plus facilement la *NP*-complétude d'un autre problème. Afin de montrer qu'un certain problème de décision $\mathcal{P}$ est *NP*-complet, il nous suffira de prouver que $\mathcal{P} \in NP$ et que le PROBLÈME DE LA SATISFAISABILITÉ (ou tout autre problème que nous savons déjà être *NP*-complet) se transforme polynomialement en $\mathcal{P}$. Cela sera suffisant, puisque la transformabilité polynomiale est transitive.

La restriction suivante du PROBLÈME DE LA SATISFAISABILITÉ se révélera très utile dans plusieurs preuves de *NP*-complétude :

PROBLÈME 3SAT	
Instance	Un ensemble X de variables et une famille $\mathcal{Z}$ de clauses sur X, qui contiennent chacune exactement trois littéraux.
Question	$\mathcal{Z}$ est-il satisfaisable ?

Afin de prouver que le PROBLÈME 3SAT est *NP*-complet, nous remarquons d'abord que toute clause peut être remplacée de manière équivalente par un ensemble de 3SAT-clauses :

Proposition 15.21. *Soit X un ensemble de variables et Z une clause sur X constituée de k littéraux. Il existe alors un ensemble Y d'au plus $\max\{k-3, 2\}$ nouvelles variables et une famille $\mathcal{Z}'$ d'au plus $\max\{k-2, 4\}$ clauses sur $X \dot\cup Y$, telle que chaque élément de $\mathcal{Z}'$ contienne exactement trois littéraux et telle que, pour toute famille $\mathcal{W}$ de clauses sur X, $\mathcal{W} \cup \{Z\}$ est satisfaisable si et seulement si $\mathcal{W} \cup \mathcal{Z}'$ est satisfaisable. De plus, une telle famille $\mathcal{Z}'$ peut être calculée en temps $O(k)$.*

Preuve. Si Z est constituée de trois littéraux, on pose $\mathcal{Z}' := \{Z\}$. Si Z contient plus de trois littéraux, disons $Z = \{\lambda_1, \ldots, \lambda_k\}$, nous choisissons un ensemble $Y = \{y_1, \ldots, y_{k-3}\}$ de $k-3$ nouvelles variables et nous posons

$$\mathcal{Z}' := \big\{\{\lambda_1, \lambda_2, y_1\}\{\overline{y_1}, \lambda_3, y_2\}, \{\overline{y_2}, \lambda_4, y_3\}, \ldots, \{\overline{y_{k-4}}, \lambda_{k-2}, y_{k-3}\}, \{\overline{y_{k-3}}, \lambda_{k-1}, \lambda_k\}\big\}.$$

Si $Z = \{\lambda_1, \lambda_2\}$, nous choisissons une nouvelle variable y_1 ($Y := \{y_1\}$) et posons

$$\mathcal{Z}' := \{\{\lambda_1, \lambda_2, y_1\}, \{\lambda_1, \lambda_2, \overline{y_1}\}\}.$$

Si $Z = \{\lambda_1\}$, nous choisissons un ensemble $Y = \{y_1, y_2\}$ de deux nouvelles variables et posons

$$\mathcal{Z}' := \{\{\lambda_1, y_1, y_2\}, \{\lambda_1, y_1, \overline{y_2}\}, \{\lambda_1, \overline{y_1}, y_2\}, \{\lambda_1, \overline{y_1}, \overline{y_2}\}\}.$$

Observons que, dans tous les cas, Z peut être remplacée de manière équivalente par $\mathcal{Z}'$ dans toute instance du PROBLÈME DE LA SATISFAISABILITÉ. □

Théorème 15.22. (Cook [1971]) *Le* PROBLÈME 3SAT *est NP-complet.*

Preuve. Le problème 3SAT est certainement dans *NP*, puisqu'il est une restriction du PROBLÈME DE LA SATISFAISABILITÉ. Nous montrons maintenant que le PROBLÈME DE LA SATISFAISABILITÉ se ramène polynomialement au PROBLÈME 3SAT. Considérons une collection $\mathcal{Z}$ de clauses $Z_1, \ldots, Z_m$. Nous allons construire une nouvelle famille $\mathcal{Z}'$ de clauses, contenant chacune trois littéraux, telle que $\mathcal{Z}$ soit satisfaisable si et seulement si $\mathcal{Z}'$ est satisfaisable.

Pour ce faire, nous remplaçons chaque clause Z_i par un ensemble de clauses équivalent, chaque clause contenant trois littéraux. Cela est possible en temps linéaire d'après la proposition 15.21. □

Si chaque clause est restreinte à seulement deux littéraux, le problème (appelé 2SAT) peut être résolu en temps linéaire (exercice 7).

15.5 Quelques problèmes *NP*-complets de base

Karp a découvert l'importance des conséquences du travail de Cook pour les problèmes d'optimisation combinatoire. Pour commencer, considérons le problème suivant :

PROBLÈME DE L'ENSEMBLE STABLE	
Instance	Un graphe G et un entier k.
Question	Existe-t-il un ensemble stable de k sommets ?

Théorème 15.23. (Karp [1972]) *Le* PROBLÈME DE L'ENSEMBLE STABLE *est NP-complet.*

Preuve. Évidemment, le PROBLÈME DE L'ENSEMBLE STABLE appartient à *NP*. Nous montrons que le PROBLÈME DE LA SATISFAISABILITÉ se ramène polynomialement au PROBLÈME DE L'ENSEMBLE STABLE.

Soit $\mathcal{Z}$ une collection de clauses $Z_1, \ldots, Z_m$ telles que $Z_i = \{\lambda_{i1}, \ldots, \lambda_{ik_i}\}$ ($i = 1, \ldots, m$), où les λ_{ij} sont des littéraux sur un ensemble X de variables.

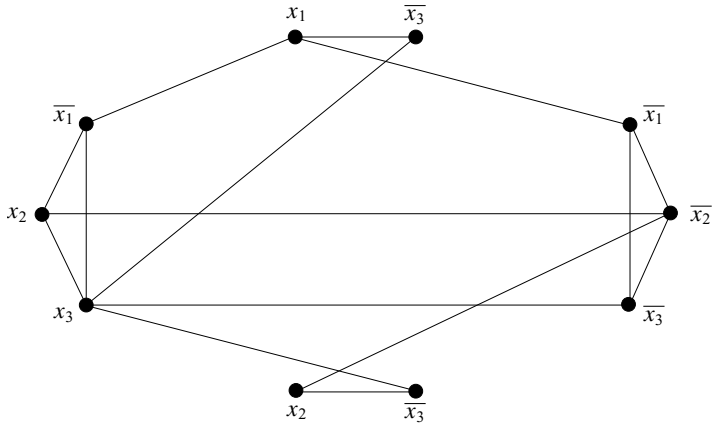


Figure 15.1.

Nous allons construire un graphe G tel que G ait un ensemble stable de taille m si et seulement s'il existe un assignement satisfaisant les m clauses.

Pour chaque clause Z_i, nous considérons une clique sur k_i sommets correspondant aux littéraux de cette clause. Les sommets associés à des clauses différentes sont connectés par une arête si et seulement si les littéraux correspondants se contredisent l'un l'autre. Formellement, considérons $V(G) := \{v_{ij} : 1 \leq i \leq m, 1 \leq j \leq k_i\}$ et

$$E(G) := \big\{(v_{ij}, v_{kl}) : (i = k \text{ et } j \neq l) \\ \text{ou } (\lambda_{ij} = x \text{ et } \lambda_{kl} = \overline{x} \text{ pour un } x \in X)\big\}.$$

Voir la figure 15.1 pour un exemple ($m = 4$, $Z_1 = \{\overline{x_1}, x_2, x_3\}$, $Z_2 = \{x_1, \overline{x_3}\}$, $Z_3 = \{x_2, \overline{x_3}\}$ et $Z_4 = \{\overline{x_1}, \overline{x_2}, \overline{x_3}\}$).

Supposons que G possède un ensemble stable de taille m. Alors les sommets de cet ensemble stable coïncident avec des littéraux compatibles deux à deux et appartenant à des clauses différentes. En fixant chacun de ces littéraux à *vrai* et en fixant arbitrairement les variables qui n'apparaissent pas ici, on obtient un assignement satisfaisant les m clauses.

Inversement, si un assignement satisfait les m clauses, on choisit alors un littéral qui est *vrai* dans chacune des clauses. L'ensemble de sommets associés définit alors un ensemble stable de taille m dans G. □

Il est essentiel que k fasse partie de l'input : pour tout k fixé, on peut décider en temps $O(n^k)$ si un graphe donné sur n sommets possède un ensemble stable de taille k (en testant simplement tous les ensembles de sommets à k éléments). Nous présentons maintenant deux problèmes liés intéressants :

PROBLÈME DE LA COUVERTURE PAR LES SOMMETS	
Instance	Un graphe G et un entier k.
Question	Existe-t-il une couverture par les sommets de cardinalité k ?

PROBLÈME DE LA CLIQUE	
Instance	Un graphe G et un entier k.
Question	G possède-t-il une clique de cardinalité k ?

Corollaire 15.24. (Karp [1972]) *Le* PROBLÈME DE LA COUVERTURE PAR LES SOMMETS *et le* PROBLÈME DE LA CLIQUE *sont NP-complets.*

Preuve. D'après la proposition 2.2, le PROBLÈME DE L'ENSEMBLE STABLE se transforme polynomialement en problèmes de la COUVERTURE PAR LES SOMMETS et de la CLIQUE. □

Nous passons maintenant au célèbre PROBLÈME DU CYCLE HAMILTONIEN (déjà défini au paragraphe 15.3).

Théorème 15.25. (Karp [1972]) *Le* PROBLÈME DU CYCLE HAMILTONIEN *est NP-complet.*

Preuve. L'appartenance à *NP* est évidente. Nous prouvons que le PROBLÈME 3SAT se ramène polynomialement au PROBLÈME DU CYCLE HAMILTONIEN. Étant donné une famille $\mathcal{Z}$ de clauses $Z_1, \ldots, Z_m$ sur $X = \{x_1, \ldots, x_n\}$, chaque clause contenant trois littéraux, nous allons construire un graphe G tel que G soit hamiltonien si et seulement si $\mathcal{Z}$ est satisfaisable.

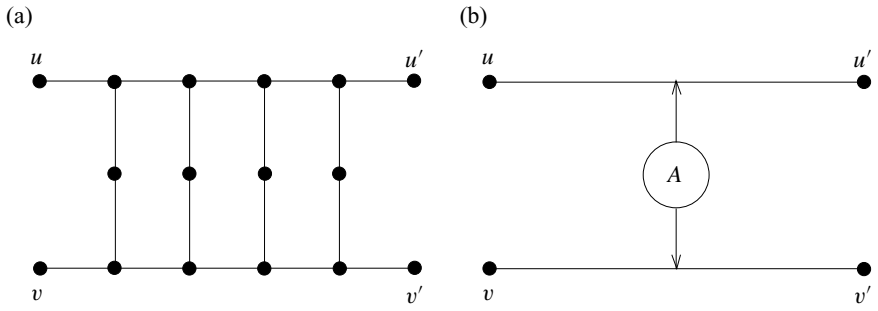


Figure 15.2.

Nous définissons d'abord deux structures qui apparaîtront plusieurs fois dans G. Considérons le graphe de la figure 15.2(a), que nous appelons A. Nous supposons que c'est un sous-graphe de G et qu'aucun sommet de A excepté u, u', v, v' n'est

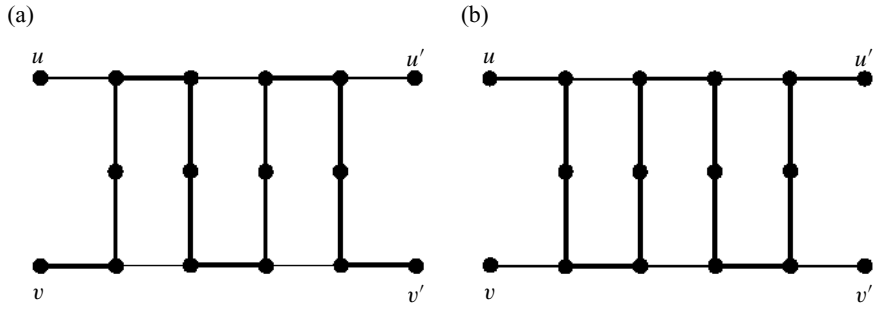


Figure 15.3.

incident à une autre arête de G. Alors tout cycle hamiltonien de G doit traverser A de l'une des façons représentées aux figures 15.3(a) et (b). On peut donc remplacer A par deux arêtes avec la restriction supplémentaire que tout cycle hamiltonien de G doit contenir exactement l'une d'entre elles (figure 15.2(b)).

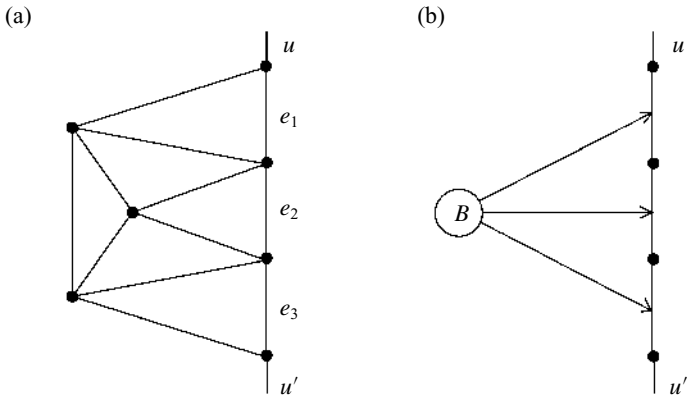


Figure 15.4.

Considérons maintenant le graphe B de la figure 15.4(a). Nous supposons que c'est un sous-graphe de G et qu'aucun sommet de B excepté u et u' n'est incident à une autre arête de G. Alors aucun cycle hamiltonien de G ne peut traverser à la fois e_1, e_2 et e_3. De plus, on vérifie facilement que pour tout $S \subset \{e_1, e_2, e_3\}$, il existe une chaîne hamiltonienne de u à u' dans B qui contient S mais pas $\{e_1, e_2, e_3\} \setminus S$. Nous représentons B par le symbole de la figure 15.4(b).

Nous pouvons maintenant construire G. Pour chaque clause, nous créons une copie de B, et chaque nouvelle copie est jointe à la précédente. Entre la première et la dernière copie de B, nous ajoutons deux sommets pour chaque variable et nous joignons tous ces sommets les uns après les autres. Ensuite, nous dédoublons

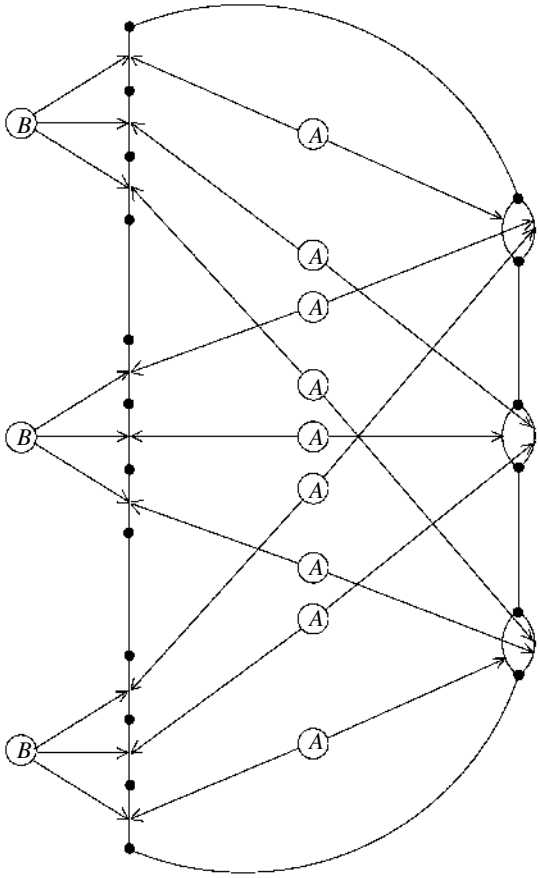


Figure 15.5.

les arêtes reliant les deux sommets associés à chaque variable x ; ces deux arêtes correspondront à x et $\overline{x}$, respectivement.

Les arêtes e_1, e_2, et e_3 de chaque copie de B sont maintenant connectées par l'intermédiaire d'une copie de A aux arêtes correspondant respectivement au premier, deuxième, et troisième littéral de la clause correspondante. Ces constructions sont faites consécutivement : lorsque l'on introduit une copie du sous-graphe A associée à une arête $e = (u, v)$ correspondant à un littéral, l'arête incidente à u de la figure 15.2(a) prend le rôle de e : c'est maintenant l'arête correspondant à ce littéral. La construction globale est illustrée par la figure 15.5 avec l'exemple $\{\{x_1, \overline{x_2}, \overline{x_3}\}, \{\overline{x_1}, x_2, \overline{x_3}\}, \{\overline{x_1}, \overline{x_2}, x_3\}\}$.

Nous affirmons maintenant que G est hamiltonien si et seulement si $\mathcal{Z}$ est satisfaisable. Soit C un cycle hamiltonien. On définit un assignement en fixant un littéral à *vrai* si et seulement si C contient l'arête correspondante. D'après les propriétés des structures A et B, chaque clause contient un littéral qui est *vrai*.

Inversement, tout assignement satisfaisant définit un ensemble d'arêtes qui correspondent à des littéraux qui sont *vrai*. Comme chaque clause contient un littéral qui est *vrai*, cet ensemble d'arêtes peut être complété en un cycle hamiltonien de G. □

Cette preuve est essentiellement due à Papadimitriou et Steiglitz [1982]. Le problème consistant à décider si un graphe donné contient une chaîne hamiltonienne est également *NP*-complet (exercice 14(a)). De plus, on peut facilement passer de la version non orientée à la version orientée du PROBLÈME DU CYCLE HAMILTONIEN ou de la chaîne hamiltonienne, en remplaçant chaque arête par une paire d'arcs opposés. Ainsi, les versions orientées (circuit hamiltonien et chemin hamiltonien) sont également des problèmes *NP*-complets.

Voici un autre problème *NP*-complet fondamental :

PROBLÈME DU COUPLAGE 3-DIMENSIONNEL (3DM)

Instance Des ensembles disjoints U, V, W de même cardinalité et $T \subseteq U \times V \times W$.

Question Existe-t-il un sous-ensemble M de T de cardinalité $|M| = |U|$ tel que, pour tout $(u, v, w), (u', v', w') \in M$ distincts, on ait $u \neq u'$, $v \neq v'$ et $w \neq w'$?

Théorème 15.26. (Karp [1972]) 3DM *est un problème NP-complet.*

Preuve. L'appartenance à *NP* est évidente. Nous allons ramener par une transformation polynomiale le PROBLÈME DE LA SATISFAISABILITÉ au problème 3DM. Étant donné une famille $\mathcal{Z}$ de clauses $Z_1, \ldots, Z_m$ sur $X = \{x_1, \ldots, x_n\}$, nous construisons une instance (U, V, W, T) du problème 3DM qui est une instance-«oui» si et seulement si $\mathcal{Z}$ est satisfaisable.

Nous définissons :

$$
\begin{aligned}
U &:= \{x_i^j, \overline{x_i}^j : i = 1, \ldots, n;\ j = 1, \ldots, m\} \\
V &:= \{a_i^j : i = 1, \ldots, n;\ j = 1, \ldots, m\} \cup \{v^j : j = 1, \ldots, m\} \\
&\qquad \cup \{c_k^j : k = 1, \ldots, n-1;\ j = 1, \ldots, m\} \\
W &:= \{b_i^j : i = 1, \ldots, n;\ j = 1, \ldots, m\} \cup \{w^j : j = 1, \ldots, m\} \\
&\qquad \cup \{d_k^j : k = 1, \ldots, n-1;\ j = 1, \ldots, m\} \\
T_1 &:= \{(x_i^j, a_i^j, b_i^j), (\overline{x_i}^j, a_i^{j+1}, b_i^j) : i = 1, \ldots, n;\ j = 1, \ldots, m\}, \\
&\qquad \text{où } a_i^{m+1} := a_i^1 \\
T_2 &:= \{(x_i^j, v^j, w^j) : i = 1, \ldots, n;\ j = 1, \ldots, m;\ x_i \in Z_j\} \\
&\qquad \cup \{(\overline{x_i}^j, v^j, w^j) : i = 1, \ldots, n;\ j = 1, \ldots, m;\ \overline{x_i} \in Z_j\} \\
T_3 &:= \{(x_i^j, c_k^j, d_k^j), (\overline{x_i}^j, c_k^j, d_k^j) : i = 1, \ldots, n;\ j = 1, \ldots, m;\ k = 1, \ldots, n-1\} \\
T &:= T_1 \cup T_2 \cup T_3.
\end{aligned}
$$

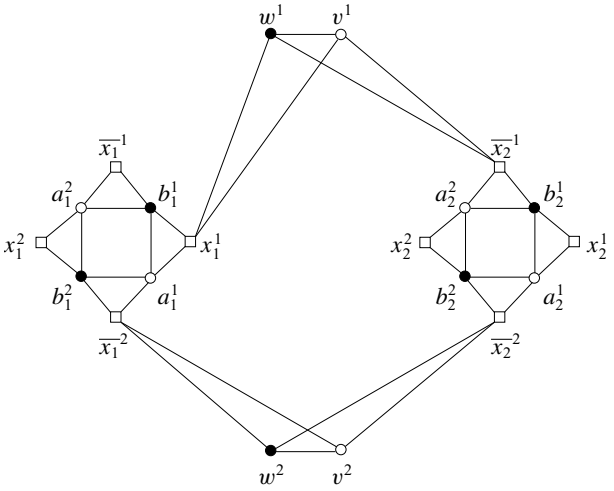


Figure 15.6.

Pour une illustration de cette construction, voir la figure 15.6. Ici $m = 2$, $Z_1 = \{x_1, \overline{x_2}\}$, $Z_2 = \{\overline{x_1}, \overline{x_2}\}$. Chaque triangle correspond à un élément de $T_1 \cup T_2$. Les éléments c_k^j, d_k^j et les triplets de T_3 ne sont pas représentés.

Supposons que (U, V, W, T) soit une instance-«oui» et soit donc $M \subseteq T$ une solution. Comme les a_i^j et les b_i^j apparaissent seulement parmi les éléments de T_1 pour chaque i, on a, soit $M \cap T_1 \supseteq \{(x_i^j, a_i^j, b_i^j) : j = 1, \ldots, m\}$, soit $M \cap T_1 \supseteq \{(\overline{x_i}^j, a_i^{j+1}, b_i^j) : j = 1, \ldots, m\}$. Dans le premier cas, nous mettons x_i à *faux*, dans le second cas à *vrai*.

De plus, pour chaque clause Z_j, on a $(\lambda^j, v^j, w^j) \in M$ pour un certain littéral $\lambda \in Z_j$. Ce littéral est *vrai* puisque λ^j n'apparaît dans aucun élément de $M \cap T_1$; nous avons ainsi un assignement satisfaisant.

Inversement, un assignement satisfaisant suggère un ensemble $M_1 \subseteq T_1$ de cardinalité nm et un ensemble $M_2 \subseteq T_2$ de cardinalité m tels que, pour toute paire d'éléments distincts $(u, v, w), (u', v', w')$ de $M_1 \cup M_2$, on ait $u \neq u'$, $v \neq v'$ et $w \neq w'$. Il est facile de compléter $M_1 \cup M_2$ par $(n-1)m$ éléments de T_3 pour obtenir une solution de l'instance du problème 3DM. □

Le problème suivant peut paraître simple, mais on ne sait pas s'il peut être résolu en temps polynomial :

PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES	
Instance	Des nombres entiers $c_1, \ldots, c_n, K$.
Question	Existe-t-il un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} c_j = K$?

Corollaire 15.27. (Karp [1972]) *Le* PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES *est NP-complet.*

Preuve. Il est évident que le PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES appartient à *NP*. Prouvons que le problème 3DM se ramène polynomialement au PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES. Soit donc (U, V, W, T) une instance du problème 3DM. Sans perte de généralité, posons $U \cup V \cup W = \{u_1, \ldots, u_{3m}\}$. Posons $S := \{\{a, b, c\} : (a, b, c) \in T\}$ et $S = \{s_1, \ldots, s_n\}$. Définissons

$$c_j := \sum_{u_i \in s_j} (n+1)^{i-1} \qquad (j = 1, \ldots, n)$$

et

$$K := \sum_{i=1}^{3m} (n+1)^{i-1}.$$

Écrit sous forme $(n+1)$-aire, le nombre c_j peut être considéré comme le vecteur d'incidence de s_j $(j = 1, \ldots, n)$, et K est représenté uniquement par des 1. Ainsi chaque solution de l'instance du problème 3DM correspond à un sous-ensemble R de S tel que $\sum_{s_j \in R} c_j = K$, et vice versa. De plus, $\text{taille}(c_j) \leq \text{taille}(K) = O(m \log n)$, la transformation polynomiale précédente est donc bien polynomiale. □

Le problème suivant est un cas particulier important :

PROBLÈME DE LA PARTITION	
Instance	Nombres entiers $c_1, \ldots, c_n$.
Question	Existe-t-il un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$?

Corollaire 15.28. (Karp [1972]) *Le* PROBLÈME DE LA PARTITION *est NP-complet.*

Preuve. Nous montrons que le PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES se ramène polynomialement au PROBLÈME DE LA PARTITION. Soit donc une instance $c_1, \ldots, c_n, K$ du PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES. On ajoute un élément $c_{n+1} := |\sum_{i=1}^{n} c_i - 2K|$ (sauf si ce nombre vaut zéro) et on a une instance $c_1, \ldots, c_{n+1}$ du PROBLÈME DE LA PARTITION.

Cas 1 : $2K \leq \sum_{i=1}^{n} c_i$. Alors, pour tout $I \subseteq \{1, \ldots, n\}$, on a

$$\sum_{i \in I} c_i = K \quad \text{si et seulement si} \quad \sum_{i \in I \cup \{n+1\}} c_i = \sum_{i \in \{1, \ldots, n\} \setminus I} c_i.$$

Cas 2 : $2K > \sum_{i=1}^{n} c_i$. Alors, pour tout $I \subseteq \{1, \ldots, n\}$, on a

$$\sum_{i \in I} c_i = K \quad \text{si et seulement si} \quad \sum_{i \in I} c_i = \sum_{i \in \{1, \ldots, n+1\} \setminus I} c_i.$$

Dans les deux cas, on a construit une instance-«oui» du PROBLÈME DE LA PARTITION si et seulement si l'instance de départ du PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES est une instance-«oui». □

Nous observons finalement :

Théorème 15.29. *Le problème de décision associé aux* INÉGALITÉS LINÉAIRES EN NOMBRES ENTIERS *est NP-complet.*

Preuve. À la proposition 15.12, nous avons déjà mentionné l'appartenance à *NP*. Chacun des problèmes précédents peut facilement être formulé comme une instance du problème de décision associé à des INÉGALITÉS LINÉAIRES EN NOMBRES ENTIERS. Par exemple, une instance du PROBLÈME DE LA PARTITION $c_1, \ldots, c_n$ est une instance-«oui» si et seulement si $\{x \in \mathbb{Z}^n : 0 \leq x \leq \mathbb{1},\ 2c^\top x = c^\top \mathbb{1}\}$ est non vide. □

15.6 Classe *coNP*

La définition de la classe *NP* n'implique pas de relation de symétrie entre les instances-«oui» et les instances-«non». Par exemple, l'appartenance à *NP* du problème suivant est une question ouverte : étant donné un graphe G, est-il vrai que G n'est pas hamiltonien ? Nous introduisons les définitions suivantes :

Définition 15.30. *Pour un problème de décision $\mathcal{P} = (X, Y)$, son* **complémentaire** *est défini par le problème de décision $(X, X \setminus Y)$. La classe coNP est constituée de tous les problèmes dont les complémentaires sont dans NP. Un problème de décision $\mathcal{P} \in$ coNP est dit* ***coNP*-complet** *si tout autre problème de coNP se transforme polynomialement en $\mathcal{P}$.*

Évidemment, le complémentaire d'un problème dans *P* est aussi dans *P*. D'autre part, *NP* $\neq$ *coNP* est communément conjecturé (bien que ce ne soit pas prouvé). Les problèmes *NP*-complets jouent un rôle particulier par rapport à cette conjecture :

Théorème 15.31. *Un problème de décision est coNP-complet si et seulement si son complémentaire est NP-complet. À moins que NP = coNP, aucun problème coNP-complet n'est dans NP.*

Preuve. La première affirmation est une conséquence directe de la définition.

Supposons que $\mathcal{P} = (X, Y) \in$ *NP* soit un problème *coNP*-complet. Soit $\mathcal{Q} = (V, W)$ un problème quelconque de *coNP*. Nous montrons que $\mathcal{Q} \in$ *NP*.

Comme $\mathcal{P}$ est *coNP*-complet, $\mathcal{Q}$ se transforme polynomialement en $\mathcal{P}$. Il existe donc un algorithme polynomial qui permet de transformer toute instance v de $\mathcal{Q}$ en une instance $x = f(v)$ de $\mathcal{P}$ telle que $x \in Y$ si et seulement si $v \in W$. Remarquons que $\text{taille}(x) \leq p(\text{taille}(v))$ pour un certain polynôme p fixé.

Comme $\mathcal{P} \in$ *NP*, il existe un polynôme q et un problème de décision $\mathcal{P}' = (X', Y')$ dans *P*, où $X' := \left\{x \# c : x \in X,\ c \in \{0,1\}^{\lfloor q(\text{taille}(x))\rfloor}\right\}$, tels que

$$Y = \left\{y \in X : \text{Il existe une chaîne } c \in \{0,1\}^{\lfloor q(\text{taille}(y))\rfloor} \text{ telle que } y \# c \in Y'\right\}$$

(voir définition 15.9). On définit un problème de décision (V', W') par $V' := \left\{v\#c : v \in V, c \in \{0,1\}^{\lfloor q(p(\text{taille}(v)))\rfloor}\right\}$, et $v\#c \in W'$ si et seulement si $f(v)\#c' \in Y'$ où c' est constitué des $\lfloor q(\text{taille}(f(v)))\rfloor$ premières composantes de c.

Observons que $(V', W') \in P$. Ainsi, par définition, $\mathcal{Q} \in NP$. Nous en concluons que $coNP \subseteq NP$ et alors, par symétrie, $NP = coNP$. □

Si on peut montrer qu'un problème est dans $NP \cap coNP$, on dit que le problème possède une **bonne caractérisation** (Edmonds [1965]). Cela signifie qu'il existe, aussi bien pour les instances-«oui» que pour les instances-«non», des certificats qui peuvent être vérifiés en temps polynomial. Le théorème 15.31 indique qu'un problème, ayant une bonne caractérisation, n'est probablement pas *NP*-complet.

Pour donner quelques exemples, la proposition 2.9, le théorème 2.24 et la proposition 2.27 fournissent de bonnes caractérisations pour les problèmes consistant à décider respectivement si un graphe donné est sans circuit, s'il possède un parcours eulérien et s'il est biparti. Bien entendu, cela n'est pas très intéressant puisque tous ces problèmes peuvent être résolus facilement en temps polynomial. Mais considérons le problème de décision associé à la PROGRAMMATION LINÉAIRE :

Théorème 15.32. *Le problème associé aux* INÉGALITÉS LINÉAIRES *est dans NP*∩ *coNP.*

Preuve. C'est une conséquence immédiate du théorème 4.4 et du corollaire 3.24. □

Bien entendu, ce théorème est aussi la conséquence de l'existence d'algorithmes polynomiaux pour la PROGRAMMATION LINÉAIRE, (par le théorème 4.18 par exemple). Cependant, avant que l'on ne découvre la MÉTHODE DES ELLIPSOÏDES, le théorème 15.32 était la seule preuve théorique que le problème associé aux INÉGALITÉS LINÉAIRES n'est probablement pas *NP*-complet. Cela donna l'espoir de trouver un algorithme polynomial pour la PROGRAMMATION LINÉAIRE (qui peut être réduit au problème associé aux INÉGALITÉS LINÉAIRES d'après la proposition 4.16) ; espoir justifié comme nous le savons aujourd'hui.

Le problème bien connu suivant a une histoire similaire :

PROBLÈME DU NOMBRE PREMIER	
Instance	Un nombre $n \in \mathbb{N}$ (dans sa représentation binaire).
Question	n est-il un nombre premier ?

Il est évident que le PROBLÈME DU NOMBRE PREMIER appartient à *coNP*. Pratt [1975] a prouvé que le PROBLÈME DU NOMBRE PREMIER appartient également à *NP*. Finalement, Agrawal, Kayal et Saxena [2004] ont prouvé que le PROBLÈME DU NOMBRE PREMIER $\in P$ en trouvant un algorithme étonnamment simple en $O(\log^{7.5+\epsilon} n)$ (pour tout $\epsilon > 0$). Auparavant, l'algorithme déterministe le plus connu pour le PROBLÈME DU NOMBRE PREMIER était dû à Adleman, Pomerance et Rumely [1983], et son temps de calcul était en $O\left((\log n)^{c \log\log\log n}\right)$ pour une

certaine constante c. Puisque la taille de l'input est $O(\log n)$, cet algorithme n'est pas polynomial.

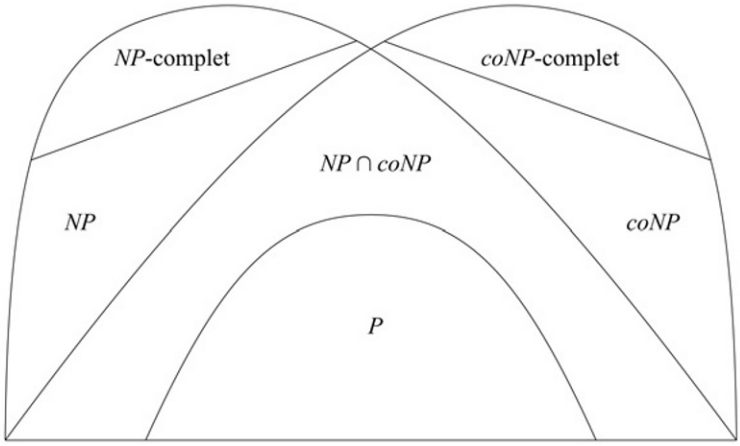


Figure 15.7.

Nous terminons ce paragraphe en esquissant les inclusions de *NP* et *coNP* (figure 15.7). Ladner [1975] a montré que, à moins que $P = NP$, il existe des problèmes dans $NP \setminus P$ qui ne sont pas *NP*-complets. Cependant, tant que la conjecture $P \neq NP$ n'est pas résolue, il est encore possible que toutes les régions, représentées sur la figure 15.7, se réduisent à une seule.

15.7 Problèmes *NP*-difficiles

Nous étendons maintenant nos résultats aux problèmes d'optimisation. Nous commençons par définir formellement le type de problème d'optimisation qui nous intéresse :

Définition 15.33. *Un* **problème d'optimisation (discrète)** *est un quadruplet* $\mathcal{P} = (X, (S_x)_{x \in X}, c, \text{but})$*, où :*

- X *est un langage sur* $\{0,1\}$ *décidable en temps polynomial ;*
- S_x *est un sous-ensemble de* $\{0,1\}^*$ *pour chaque* $x \in X$ *; il existe un polynôme* p *tel que* $\text{taille}(y) \leq p(\text{taille}(x))$ *pour tout* $y \in S_x$ *et tout* $x \in X$*, et les langages* $\{(x,y) : x \in X,\ y \in S_x\}$ *et* $\{x \in X : S_x = \emptyset\}$ *sont décidables en temps polynomial ;*
- $c : \{(x,y) : x \in X,\ y \in S_x\} \to \mathbb{Q}$ *est une fonction calculable en temps polynomial ; et*
- $\text{but} \in \{\max, \min\}$.

Les éléments de X *sont appelés des* **instances** *de* $\mathcal{P}$*. Pour chaque instance* x*, les éléments de* S_x *sont appelés les* **solutions réalisables** *de* x*. Nous notons*

$\mathrm{OPT}(x) := \mathrm{but}\{c(x,y) : y \in S_x\}$. *Une* **solution optimale** *de x est une solution réalisable y de x telle que $c(x,y) = \mathrm{OPT}(x)$.*

Un **algorithme** *pour un problème d'optimisation $(X, (S_x)_{x\in X}, c, \mathrm{but})$ est un algorithme A qui calcule pour chaque input $x \in X$ tel que $S_x \neq \emptyset$ une solution réalisable $y \in S_x$. Nous écrivons parfois $A(x) := c(x,y)$. Si $A(x) = \mathrm{OPT}(x)$ pour tout $x \in X$ tel que $S_x \neq \emptyset$, alors A est un* **algorithme exact**.

Suivant le contexte, $c(x,y)$ est souvent appelé le coût ou le poids, le profit ou la longueur de y. Si c est non négatif, on dit alors que le problème d'optimisation a des poids non négatifs. Les valeurs de c sont des nombres rationnels ; on suppose comme d'habitude l'existence d'un codage sous forme de chaînes binaires.

Le concept de réduction polynomiale s'étend facilement aux problèmes d'optimisation : un problème se réduit polynomialement à un problème d'optimisation $\mathcal{P} = (X, (S_x)_{x\in X}, c, \mathrm{but})$ s'il a un algorithme polynomial exact fondé sur un oracle, utilisant n'importe quelle fonction f telle que $f(x) \in \{y \in S_x : c(x,y) = \mathrm{OPT}(x)\}$ pour tout $x \in X$ tel que $S_x \neq \emptyset$. Nous pouvons maintenant définir :

Définition 15.34. *Un problème d'optimisation ou un problème de décision $\mathcal{P}$ est dit* ***NP*-difficile** *si tous les problèmes dans NP se réduisent polynomialement à $\mathcal{P}$.*

Remarquons que cette définition est symétrique : un problème de décision est *NP*-difficile si et seulement si son complémentaire l'est. Les problèmes *NP*-difficiles sont au moins aussi difficiles que les plus difficiles problèmes de *NP*. Mais certains peuvent être plus difficiles que tout problème dans *NP*. Un problème qui se réduit polynomialement à un certain problème dans *NP* est dit ***NP*-facile**. Un problème qui est à la fois *NP*-difficile et *NP*-facile est ***NP*-équivalent**. En d'autres termes, un problème est *NP*-équivalent si et seulement s'il est polynomialement équivalent au PROBLÈME DE LA SATISFAISABILITÉ, où deux problèmes $\mathcal{P}$ et $\mathcal{Q}$ sont dits **polynomialement équivalents** si $\mathcal{P}$ se réduit polynomialement à $\mathcal{Q}$, et $\mathcal{Q}$ se réduit polynomialement à $\mathcal{P}$. Nous avons le résultat suivant :

Proposition 15.35. *Soit $\mathcal{P}$ un problème NP-équivalent. Alors $\mathcal{P}$ a un algorithme polynomial exact si et seulement si P = NP.* □

Bien entendu, tous les problèmes *NP*-complets et tous les problèmes *coNP*-complets sont *NP*-équivalents. La plupart des problèmes présentés dans ce livre sont *NP*-faciles, puisqu'ils se réduisent polynomialement à la PROGRAMMATION EN NOMBRES ENTIERS ; c'est en général une observation évidente qui n'est même pas mentionnée. D'autre part, la plupart des problèmes que nous avons présentés jusqu'à présent sont aussi *NP*-difficiles, et nous prouverons généralement cela en décrivant une réduction polynomiale à partir d'un problème *NP*-complet. Nous allons considérer comme premier exemple le problème MAX-2SAT : étant donné une instance du PROBLÈME DE LA SATISFAISABILITÉ où chaque clause est constituée d'exactement deux littéraux, trouver un assignement qui maximise le nombre de clauses satisfaites.

Théorème 15.36. (Garey, Johnson et Stockmeyer [1976]) *Le problème* MAX-2SAT *est NP-difficile.*

Preuve. Par réduction à partir du problème 3SAT. Étant donné une instance I du problème 3SAT avec pour clauses $C_1, \ldots, C_m$, nous construisons une instance I' du problème MAX-2SAT en ajoutant de nouvelles variables $y_1, z_1, \ldots, y_m, z_m$ et en remplaçant chaque clause $C_i = \{\lambda_1, \lambda_2, \lambda_3\}$ par les quatorze clauses

$$\{\lambda_1, z_i\}, \{\lambda_1, \bar{z}_i\}, \{\lambda_2, z_i\}, \{\lambda_2, \bar{z}_i\}, \{\lambda_3, z_i\}, \{\lambda_3, \bar{z}_i\}, \{y_i, z_i\}, \{y_i, \bar{z}_i\},$$
$$\{\lambda_1, \bar{y}_i\}, \{\lambda_2, \bar{y}_i\}, \{\lambda_3, \bar{y}_i\}, \{\bar{\lambda}_1, \bar{\lambda}_2\}, \{\bar{\lambda}_1, \bar{\lambda}_3\}, \{\bar{\lambda}_2, \bar{\lambda}_3\}.$$

Remarquons qu'aucun assignement ne peut satisfaire plus de 11 de ces 14 clauses. De plus, si 11 de ces clauses sont satisfaites, alors au moins l'un des littéraux $\lambda_1, \lambda_2, \lambda_3$ doit être *vrai*. D'autre part, si un des littéraux $\lambda_1, \lambda_2, \lambda_3$ est *vrai* on peut poser $y_i := \lambda_1 \wedge \lambda_2 \wedge \lambda_3$ et $z_i :=$ *vrai* afin de satisfaire 11 de ces clauses.

Nous en concluons que l'instance I a un assignement qui satisfait les m clauses si et seulement si l'instance I' a un assignement qui satisfait $11m$ clauses. □

Savoir si tout problème de décision *NP*-difficile, $\mathcal{P} \in$ *NP*, est *NP*-complet est une question ouverte (rappelons la différence entre réduction polynomiale et transformation polynomiale ; définitions 15.15 et 15.17). Les exercices 17 et 18 présentent deux problèmes *NP*-difficiles qui ne sont pas dans *NP*.

Il n'existe pas d'algorithme polynomial pour résoudre un problème *NP*-difficile, à moins que *P* = *NP*. Cependant, il peut exister un algorithme pseudo-polynomial.

Définition 15.37. *Soit $\mathcal{P}$ un problème de décision ou un problème d'optimisation tel que chaque instance x soit constituée d'une liste d'entiers. Notons* $\text{large}(x)$ *le plus grand de ces entiers. Un algorithme pour $\mathcal{P}$ est dit* **pseudo-polynomial** *si son temps de calcul est borné par un polynôme par rapport à* $\text{taille}(x)$ *et* $\text{large}(x)$.

Par exemple, il existe un algorithme pseudo-polynomial évident pour le PROBLÈME DU NOMBRE PREMIER. Cet algorithme divise le nombre entier n, dont on teste la primalité, par chaque entier compris entre 2 et $\lfloor \sqrt{n} \rfloor$. Voici un autre exemple :

Théorème 15.38. *Il existe un algorithme pseudo-polynomial pour le* PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES.

Preuve. Étant donné une instance $c_1, \ldots, c_n, K$ du PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES, on construit un graphe orienté G avec pour ensemble de sommets $\{0, \ldots, n\} \times \{0, 1, 2, \ldots, K\}$. Pour chaque $j \in \{1, \ldots, n\}$, on ajoute les arcs $((j-1, i), (j, i))$ $(i = 0, 1, \ldots, K)$ et $((j-1, i), (j, i + c_j))$ $(i = 0, 1, \ldots, K - c_j)$.

Observons que tout chemin de $(0, 0)$ à (j, i) correspond à un sous-ensemble $S \subseteq \{1, \ldots, j\}$ tel que $\sum_{k \in S} c_k = i$, et vice versa. On peut ainsi résoudre notre instance du PROBLÈME DE LA SOMME DE SOUS-ENSEMBLES en vérifiant si G contient un chemin de $(0, 0)$ à (n, K). Cela peut être fait en temps $O(nK)$ à l'aide de l'ALGORITHME DE BALAYAGE DE GRAPHES. Nous avons donc un algorithme pseudo-polynomial.

□

L'algorithme précédent est aussi un algorithme pseudo-polynomial pour le PROBLÈME DE LA PARTITION, car $\frac{1}{2}\sum_{i=1}^{n} c_i \leq \frac{n}{2}\operatorname{large}(c_1,\ldots,c_n)$. Nous présenterons une extension de cet algorithme au paragraphe 17.2. Si les nombres ne sont pas trop grands, un algorithme pseudo-polynomial peut être assez efficace. Par conséquent, la définition suivante se révéle utile :

Définition 15.39. *Pour un problème de décision $\mathcal{P} = (X, Y)$ ou un problème d'optimisation $\mathcal{P} = (X, (S_x)_{x\in X}, c, \text{but})$, et un sous-ensemble $X' \subseteq X$ d'instances, on définit la* **restriction** *de $\mathcal{P}$ à X' par $\mathcal{P}' = (X', X' \cap Y)$ ou $\mathcal{P}' = (X', (S_x)_{x\in X'}, c, \text{but})$, respectivement.*

Soit $\mathcal{P}$ un problème de décision ou d'optimisation tel que chaque instance soit constituée d'une liste d'entiers. Pour un polynôme p, soit $\mathcal{P}_p$ la restriction de $\mathcal{P}$ aux instances x telles que $\operatorname{large}(x) \leq p(\operatorname{taille}(x))$. *$\mathcal{P}$ est dit* **fortement *NP*-difficile** *s'il existe un polynôme p tel que $\mathcal{P}_p$ soit NP-difficile. $\mathcal{P}$ est dit* **fortement *NP*-complet** *si $\mathcal{P} \in$ NP et s'il existe un polynôme p tel que $\mathcal{P}_p$ soit NP-complet.*

Proposition 15.40. *Il n'existe pas d'algorithme pseudo-polynomial exact pour chaque problème fortement NP-difficile à moins que P = NP.* □

Nous donnons par la suite quelques exemples célèbres :

Théorème 15.41. *Le problème associé à la* PROGRAMMATION EN NOMBRES ENTIERS *est fortement NP-difficile.*

Preuve. Pour un graphe non orienté G, le programme en nombres entiers $\max\{\mathbb{1}x : x \in \mathbb{Z}^{V(G)},\ 0 \leq x \leq \mathbb{1},\ x_v + x_w \leq 1 \text{ pour } (v,w) \in E(G)\}$ a une valeur optimale supérieure ou égale à k si et seulement si G contient un ensemble stable de cardinalité k. Puisque $k \leq |V(G)|$ pour toutes les instances non triviales (G,k) du PROBLÈME DE L'ENSEMBLE STABLE, le résultat découle du théorème 15.23. □

PROBLÈME DU VOYAGEUR DE COMMERCE (PVC)	
Instance	Un graphe complet K_n ($n \geq 3$) et des poids $c : E(K_n) \to \mathbb{Q}_+$.
Tâche	Trouver un cycle hamiltonien T dont le poids $\sum_{e\in E(T)} c(e)$ soit minimum.

Les sommets d'une instance du PVC correspondent généralement à des villes et les poids à des distances.

Théorème 15.42. *Le* PVC *est fortement NP-difficile.*

Preuve. Nous montrons que le PVC est *NP*-difficile même si on le restreint aux instances où toutes les distances sont égales à 1 ou 2. Nous décrivons une transformation polynomiale à partir du PROBLÈME DU CYCLE HAMILTONIEN. Étant donné un graphe G sur n sommets, on construit l'instance suivante du PVC : associons une ville à chaque sommet de G, et fixons les coûts à 1 si l'arête est dans $E(G)$ et à 2 sinon. Il est alors évident que G est hamiltonien si et seulement si le cycle optimal du PVC est de longueur n. □

La preuve montre également que le problème de décision suivant n'est pas plus facile que le PVC : étant donné une instance du PVC et un entier k, existe-t-il un cycle de longueur inférieure ou égale à k ? Une affirmation semblable est vraie pour une importante classe de problèmes d'optimisation discrète :

Proposition 15.43. *Soient $\mathcal{F}$ et $\mathcal{F}'$ des familles (infinies) d'ensembles finis, et soit $\mathcal{P}$ le problème d'optimisation suivant : étant donné un ensemble $E \in \mathcal{F}$ et une fonction $c : E \to \mathbb{Z}$, trouver un ensemble $F \subseteq E$ tel que $F \in \mathcal{F}'$ et que $c(F)$ soit minimum (ou conclure qu'il n'existe pas un tel ensemble F).*

Alors $\mathcal{P}$ peut être résolu en temps polynomial si et seulement si le problème de décision suivant peut être résolu en temps polynomial : étant donné une instance (E, c) de $\mathcal{P}$ et un entier k, avons-nous $\mathrm{OPT}((E, c)) \leq k$? Si le problème d'optimisation est NP-difficile, alors le problème de décision l'est également.

Preuve. Il suffit de montrer qu'il existe un algorithme fondé sur un oracle pour le problème d'optimisation en utilisant le problème de décision (le sens contraire est évident). Soit (E, c) une instance de $\mathcal{P}$. Nous déterminons tout d'abord $\mathrm{OPT}((E, c))$ par recherche dichotomique. Puisqu'il existe au plus $1 + \sum_{e \in E} |c(e)| \leq 2^{\mathrm{taille}(c)}$ valeurs possibles, nous pouvons le faire à l'aide de $O(\mathrm{taille}(c))$ itérations, chacune incluant un appel à l'oracle.

Ensuite, on vérifie successivement pour chaque élément de E s'il existe une solution optimale ne contenant pas cet élément. On peut le faire en augmentant son poids (disons de un) et en vérifiant si cela augmente aussi la valeur d'une solution optimale. Si c'est le cas, nous conservons l'ancien poids, sinon nous augmentons effectivement le poids. Après avoir appliqué cette procédure à tous les éléments de E, les éléments dont le poids n'a pas été modifié constituent une solution optimale. □

Ce résultat s'applique à de nombreux exemples dont en particulier le PVC, le PROBLÈME DE LA CLIQUE DE POIDS MAXIMUM, le PROBLÈME DU PLUS COURT CHEMIN, le PROBLÈME DU SAC À DOS, et bien d'autres.

Exercices

1. Observer qu'il existe plus de langages que de machines de Turing. Conclure qu'il existe des langages qui ne peuvent pas être décidés à l'aide d'une machine de Turing.

   Les machines de Turing peuvent être également codées à l'aide de chaînes binaires. Considérons le célèbre PROBLÈME D'ARRÊT : étant donné deux chaînes binaires x et y, où x correspond au codage d'une machine de Turing Φ, a-t-on $\mathrm{time}(\Phi, y) < \infty$?

   Prouver que le PROBLÈME D'ARRÊT est indécidable (c.-à-d. qu'il n'existe pas d'algorithme pour le résoudre).

   Indication : en supposant qu'un tel algorithme A existe, construire une machine

de Turing qui, sur l'input x, exécute tout d'abord l'algorithme A sur l'input (x, x) et s'arrête ensuite si et seulement si $\text{output}(A, (x, x)) = 0$.

2. Décrire une machine de Turing qui compare deux chaînes : elle devrait accepter en input une chaîne $a\#b$ telle que $a, b \in \{0, 1\}^*$ et renvoyer 1 si $a = b$ et 0 si $a \neq b$.
3. Un modèle de machine bien connu est la **machine RAM** : il fonctionne avec une suite infinie de registres $x_1, x_2, \ldots$ et un registre particulier, l'accumulateur *Acc*. Chaque registre peut stocker un nombre entier arbitrairement grand, éventuellement négatif. Un programme RAM correspond à une séquence d'instructions. Il y a dix types d'instructions (leur signification est illustrée sur le côté droit) :

WRITE	k	$Acc := k$.
LOAD	k	$Acc := x_k$.
LOADI	k	$Acc := x_{x_k}$.
STORE	k	$x_k := Acc$.
STOREI	k	$x_{x_k} := Acc$.
ADD	k	$Acc := Acc + x_k$.
SUBTR	k	$Acc := Acc - x_k$.
HALF	k	$Acc := \lfloor Acc/2 \rfloor$.
IFPOS	i	**Si** $Acc > 0$ **alors aller en** ⓘ.
HALT		**Arrêter**.

Un programme RAM est une séquence de m instructions ; chacune étant d'un des types précédents, où $k \in \mathbb{Z}$ et $i \in \{1, \ldots, m\}$. Le calcul commence avec l'instruction 1 ; il s'exécute ensuite comme on peut s'y attendre ; nous ne donnons pas de définition formelle.

La liste d'instructions précédente peut être étendue. On dit qu'une commande peut être simulée par un programme RAM en temps n, si elle peut être remplacée par des commandes RAM, de telle façon que le nombre total d'étapes nécessaires à tout calcul augmente d'au plus un facteur n.

(a) Montrer que les commandes suivantes peuvent être simulées par de courts programmes RAM en temps constant :

IFNEG	k	**Si** $Acc < 0$ **alors aller en** ⓚ.
IFZERO	k	**Si** $Acc = 0$ **alors aller en** ⓚ.

∗ (b) Montrer que les commandes SUBTR et HALF peuvent être simulées par des programmes RAM utilisant seulement les huit autres commandes en temps $O(\text{taille}(x_k))$ et $O(\text{taille}(Acc))$, respectivement.

∗ (c) Montrer que les commandes suivantes peuvent être simulées par des programmes RAM en temps $O(n)$, où $n = \max\{\text{taille}(x_k), \text{taille}(Acc)\}$:

MULT	k	$Acc := Acc \cdot x_k$.
DIV	k	$Acc := \lfloor Acc/x_k \rfloor$.
MOD	k	$Acc := Acc \bmod x_k$.

* 4. Soit $f : \{0,1\}^* \to \{0,1\}^*$ une application. Montrer que s'il existe une machine de Turing Φ calculant f, alors il existe un programme RAM (voir exercice 3) tel que le calcul sur l'input x (dans *Acc*) se termine après $O(\text{taille}(x)+\text{time}(\Phi, x))$ étapes avec $Acc = f(x)$.
Montrer que s'il existe une machine RAM qui, étant donné x dans *Acc*, calcule $f(x)$ dans *Acc* en au plus $g(\text{taille}(x))$ étapes, alors il existe une machine de Turing Φ calculant f avec $\text{time}(\Phi, x) = O(g(\text{taille}(x))^3)$.

5. Prouver que les deux problèmes de décisions suivants sont dans *NP* :
   (a) Étant donné deux graphes G et H, G est-il isomorphe à un sous-graphe de H ?
   (b) Étant donné un entier naturel n (donné sous forme binaire), existe-t-il un nombre premier p tel que $n = p^p$?

6. Prouver : si $\mathcal{P} \in NP$, alors il existe un polynôme p tel que $\mathcal{P}$ peut être résolu à l'aide d'un algorithme (déterministe) de complexité $O\left(2^{p(n)}\right)$.

7. Soit $\mathcal{Z}$ une instance du problème 2SAT, c.-à-d. un ensemble de clauses sur X contenant chacune deux littéraux. Considérons un graphe orienté $G(\mathcal{Z})$ défini de la manière suivante : $V(G)$ est l'ensemble des littéraux sur X. Il existe un arc $(\lambda_1, \lambda_2) \in E(G)$ si et seulement si la clause $\left\{\overline{\lambda}_1, \lambda_2\right\}$ fait partie de $\mathcal{Z}$.
   (a) Montrer que si, pour une variable x, x et $\overline{x}$ sont dans la même composante fortement connexe de $G(\mathcal{Z})$, alors $\mathcal{Z}$ n'est pas satisfaisable.
   (b) Prouver la réciproque de (a).
   (c) Donner un algorithme linéaire pour le problème 2SAT.

8. Décrire un algorithme linéaire qui, pour une instance donnée du PROBLÈME DE LA SATISFAISABILITÉ, fournit un assignement satisfaisant au moins la moitié des clauses.

9. Considérons le problème 3-OCCURRENCE SAT, qui correspond au PROBLÈME DE LA SATISFAISABILITÉ restreint aux instances telles que chaque clause contient au plus trois littéraux et que chaque variable apparaît dans au plus trois clauses. Prouver que même cette version restreinte est un problème *NP*-complet.

10. Soit $\kappa : \{0,1\}^m \to \{0,1\}^m$ une application (pas nécessairement bijective), $m \geq 2$. Pour $x = x_1 \times \cdots \times x_n \in \{0,1\}^m \times \cdots \times \{0,1\}^m = \{0,1\}^{nm}$, définissons $\kappa(x) := \kappa(x_1) \times \cdots \times \kappa(x_n)$, et pour un problème de décision $\mathcal{P} = (X, Y)$, tel que $X \subseteq \bigcup_{n \in \mathbb{Z}_+} \{0,1\}^{nm}$, définissons $\kappa(\mathcal{P}) := (\{\kappa(x) : x \in X\}, \{\kappa(x) : x \in Y\})$. Prouver :
    (a) Pour tout codage κ et pour tout $\mathcal{P} \in NP$, on a aussi $\kappa(\mathcal{P}) \in NP$.
    (b) Si $\kappa(\mathcal{P}) \in P$ pour tout codage κ et pour tout $\mathcal{P} \in P$, alors $P = NP$.

(Papadimitriou [1994])

11. Prouver que le PROBLÈME DE L'ENSEMBLE STABLE est *NP*-complet même si on le restreint aux graphes dont le degré maximum est égal à 4.
Indication : utiliser l'exercice 9.

12. Prouver que le problème suivant, parfois appelé PROBLÈME DE L'ENSEMBLE DOMINANT, est *NP*-complet : étant donné un graphe non orienté G et un nombre $k \in \mathbb{N}$, existe-t-il un ensemble $X \subseteq V(G)$ avec $|X| \leq k$ tel que $X \cup \Gamma(X) = V(G)$?
Indication : transformation à partir du PROBLÈME DE LA COUVERTURE PAR LES SOMMETS.

13. Le PROBLÈME DE LA CLIQUE est *NP*-complet. Est-il encore *NP*-complet (à condition que $P \neq NP$) s'il est restreint aux
    (a) graphes bipartis,
    (b) graphes planaires,
    (c) graphes 2-connexes ?

14. Prouver que les problèmes suivants sont *NP*-complets :
    (a) PROBLÈME DE LA CHAÎNE HAMILTONIENNE et PROBLÈME DU CHEMIN HAMILTONIEN
    Étant donné un graphe G non orienté (resp. orienté), G contient-il une chaîne hamiltonienne (resp. un chemin hamiltonien) ?
    (b) PROBLÈME DU PLUS COURT CHEMIN
    Étant donné un graphe G, des poids $c : E(G) \to \mathbb{Z}$, deux sommets $s, t \in V(G)$, et un entier k. Existe-t-il un chemin de s à t de poids au plus égal à k ?
    (c) PROBLÈME DE L'INTERSECTION DE TROIS MATROÏDES
    Étant donné trois matroïdes $(E, \mathcal{F}_1), (E, \mathcal{F}_2), (E, \mathcal{F}_3)$ (par des oracles d'indépendance) et un nombre $k \in \mathbb{N}$, décider s'il existe un ensemble $F \in \mathcal{F}_1 \cap \mathcal{F}_2 \cap \mathcal{F}_3$ tel que $|F| \geq k$.
    (d) PROBLÈME DU POSTIER CHINOIS
    Étant donné des graphes G et H tels que $V(G) = V(H)$, des poids $c : E(H) \to \mathbb{Z}_+$ et un entier k. Existe-t-il un sous-ensemble $F \subseteq E(H)$ avec $c(F) \leq k$ tel que $(V(G), E(G) \dot{\cup} F)$ soit connexe et eulérien ?

15. Trouver un algorithme polynomial pour les problèmes de décision suivants ou prouver qu'ils sont *NP*-complets :
    (a) Étant donné un graphe non orienté G et un sous-ensemble $T \subseteq V(G)$, existe-t-il un arbre couvrant dans G tel que tous les sommets T soient des feuilles ?
    (b) Étant donné un graphe non orienté G et un sous-ensemble $T \subseteq V(G)$, existe-t-il un arbre couvrant dans G tel que toutes ses feuilles soient des éléments de T ?

(c) Étant donné un graphe orienté G, des poids $c : E(G) \to \mathbb{R}$, un ensemble $T \subseteq V(G)$ et un nombre k, existe-t-il une ramification B telle que $|\delta_B^+(x)| \leq 1$ pour tout $x \in T$ et $c(B) \geq k$?

16. Prouver que le problème de décision suivant appartient à *coNP* : étant donné une matrice $A \in \mathbb{Q}^{m \times n}$ et un vecteur $b \in \mathbb{Q}^n$, le polyèdre $\{x : Ax \leq b\}$ est-il entier ?
Indication : utiliser la proposition 3.9, le lemme 5.11 et le théorème 5.13.
Remarque : on ne sait pas si ce problème est dans *NP*.

17. Montrer que le problème suivant est *NP*-difficile (on ne sait pas s'il est dans *NP*) : étant donné une instance duPROBLÈME DE LA SATISFAISABILITÉ, est-ce que la majorité de tous les assignements satisfait toutes les clauses ?

18. Montrer que le PROBLÈME DE LA PARTITION se ramène polynomialement au problème suivant (qui est ainsi *NP*-difficile ; on ne sait pas s'il est dans *NP*) :

k-IÈME PLUS LOURD SOUS-ENSEMBLE	
Instance	Des entiers $c_1, \ldots, c_n, K, L$.
Question:	Existe-t-il K sous-ensembles distincts $S_1, \ldots, S_K \subseteq \{1, \ldots, n\}$ tels que $\sum_{j \in S_i} c_j \geq L$ pour $i = 1, \ldots, K$?

Indication : définir K et L de manière appropriée.

19. Prouver que le problème suivant appartient à *coNP* : étant donné une matrice $A \in \mathbb{Z}^{m \times n}$ et un vecteur $b \in \mathbb{Z}^m$, décider si le polyèdre $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ est entier.
Remarque : Ce problème est effectivement *coNP*-complet. Cela a été démontré par Papadimitriou et Yannakakis [1990].

Références

Littérature générale :

Aho, A.V., Hopcroft, J.E., Ullman, J.D. [1974] : The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading 1974

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M. [1999] : Complexity and Approximation : Combinatorial Optimization Problems and Their Approximability Properties. Springer, Berlin 1999

Bovet, D.B., Crescenzi, P. [1994] : Introduction to the Theory of Complexity. Prentice-Hall, New York 1994

Garey, M.R., Johnson, D.S. [1979] : Computers and Intractability : A Guide to the Theory of *NP*-Completeness. Freeman, San Francisco 1979, Chapters 1–3, 5, and 7

Horowitz, E., Sahni, S. [1978] : Fundamentals of Computer Algorithms. Computer Science Press, Potomac 1978, Chapter 11

Johnson, D.S. [1981] : The *NP*-completeness column : an ongoing guide. Journal of Algorithms starting with Vol. 4 (1981)

Karp, R.M. [1975] : On the complexity of combinatorial problems. Networks 5 (1975), 45–68

Papadimitriou, C.H. [1994] : Computational Complexity. Addison-Wesley, Reading 1994

Papadimitriou, C.H., Steiglitz, K. [1982] : Combinatorial Optimization : Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Chapters 15 and 16

Wegener, I. [2005] : Complexity Theory : Exploring the Limits of Efficient Algorithms. Springer, Berlin 2005

Références citées :

Adleman, L.M., Pomerance, C., Rumely, R.S. [1983] : On distinguishing prime numbers from composite numbers. Annals of Mathematics 117 (1983), 173–206

Agrawal, M., Kayal, N., Saxena, N. [2004] : PRIMES is in P. Annals of Mathematics 160 (2004), 781–793

Cook, S.A. [1971] : The complexity of theorem proving procedures. Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (1971), 151–158

Edmonds, J. [1965] : Minimum partition of a matroid into independent subsets. Journal of Research of the National Bureau of Standards B 69 (1965), 67–72

Fürer, M. [2007] : Faster integer mulitplication. Proceedings of the 39th ACM Symposium on Theory of Computing (2007), 57–66

Garey, M.R., Johnson, D.S., Stockmeyer, L. [1976] : Some simplified *NP*-complete graph problems. Theoretical Computer Science 1 (1976), 237–267

Hopcroft, J.E., Ullman, J.D. [1979] : Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading 1979

Karp, R.M. [1972] : Reducibility among combinatorial problems. In : Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, eds.), Plenum Press, New York 1972, pp. 85–103

Ladner, R.E. [1975] : On the structure of polynomial time reducibility. Journal of the ACM 22 (1975), 155–171

Lewis, H.R., Papadimitriou, C.H. [1981] : Elements of the Theory of Computation. Prentice-Hall, Englewood Cliffs 1981

Papadimitriou, C.H., Yannakakis, M. [1990] : On recognizing integer polyhedra. Combinatorica 10 (1990), 107–109

Pratt, V. [1975] : Every prime has a succinct certificate. SIAM Journal on Computing 4 (1975), 214–220

Schönhage, A., Strassen, V. [1971] : Schnelle Multiplikation großer Zahlen. Computing 7 (1971), 281–292

Turing, A.M. [1936] : On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society (2) 42 (1936), 230–265 and 43 (1937), 544–546

Van Emde Boas, P. [1990] : Machine models and simulations. In : Handbook of Theoretical Computer Science ; Volume A ; Algorithms and Complexity (J. Van Leeuwen, ed.), Elsevier, Amsterdam 1990, pp. 1–66

Chapitre 16

Algorithmes d'approximation

Dans ce chapitre, nous introduisons le concept important d'algorithmes d'approximation. Jusqu'ici nous avons principalement traité des problèmes polynomiaux. Dans les chapitres restants, nous indiquerons quelques stratégies pour faire face aux problèmes d'optimisation combinatoire *NP*-difficiles. Dans ce contexte, les algorithmes d'approximation doivent être mentionnés en premier lieu.

Le cas idéal est que la solution soit garantie de différer de la solution optimale uniquement par une constante :

Définition 16.1. *Un* **algorithme d'approximation absolue** *pour un problème d'optimisation $\mathcal{P}$ est un algorithme polynomial A pour $\mathcal{P}$ pour lequel il existe une constante k telle que*

$$|A(I) - \mathrm{OPT}(I)| \ \leq \ k$$

pour toutes les instances I de $\mathcal{P}$.

Malheureusement, un algorithme d'approximation absolue n'est connu que pour un nombre très restreint de problèmes *NP*-difficiles classiques. Nous présenterons deux exemples très importants, le PROBLÈME DE LA COLORATION DES ARÊTES et le PROBLÈME DE LA COLORATION DES SOMMETS dans les graphes planaires au paragraphe 16.3.

Dans la plupart des cas, nous devons nous contenter de garanties de performance relatives. Nous nous restreignons ici aux problèmes avec des poids non négatifs.

Définition 16.2. *Soit $\mathcal{P}$ un problème d'optimisation avec des poids non négatifs et $k \geq 1$. Un* **algorithme d'approximation de facteur *k*** *pour $\mathcal{P}$ est un algorithme polynomial A pour $\mathcal{P}$ tel que*

$$\frac{1}{k}\,\mathrm{OPT}(I) \ \leq \ A(I) \ \leq \ k\,\mathrm{OPT}(I)$$

pour toutes les instances I de $\mathcal{P}$. On dit aussi que A a un **rapport de performance** *(ou une* **garantie de performance***) égal à k.*

La première inégalité s'applique aux problèmes de maximisation, la seconde aux problèmes de minimisation. Remarquons que pour les instances I telles que $\mathrm{OPT}(I) = 0$, nous avons besoin d'une solution exacte. Les algorithmes d'approximation de facteur 1 sont précisément les algorithmes polynomiaux exacts. La définition précédente est parfois étendue au cas où k est une fonction de l'instance I, plutôt qu'une constante. Nous verrons un exemple au paragraphe suivant.

Au paragraphe 13.4, nous avons vu que l'ALGORITHME GLOUTON-MEILLEUR-INSÉRÉ appliqué au PROBLÈME DE MAXIMISATION pour le système d'indépendance $(E, \mathcal{F})$ a un rapport de performance égal à $\frac{1}{q(E,\mathcal{F})}$ (théorème 13.19). Dans les paragraphes et chapitres suivants, nous illustrerons les définitions précédentes et analyserons l'approximabilité de nombreux problèmes *NP*-difficiles. Nous commençons avec les problèmes de couverture.

16.1 Couverture par des ensembles

Dans ce paragraphe, nous nous concentrons sur le problème assez général suivant :

PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM	
Instance	Un système d'ensembles $(U, \mathcal{S})$ tel que $\bigcup_{S\in\mathcal{S}} S = U$, des poids $c : \mathcal{S} \to \mathbb{R}_+$.
Tâche	Trouver une **couverture par des ensembles** de $(U, \mathcal{S})$ de poids minimum, c.-à-d. une sous-famille $\mathcal{R} \subseteq \mathcal{S}$ telle que $\bigcup_{R\in\mathcal{R}} R = U$.

Lorsque $c \equiv 1$, le problème est appelé le PROBLÈME DE LA COUVERTURE MINIMUM PAR DES ENSEMBLES. Un autre cas particulier intéressant apparaît si $|\{S \in \mathcal{S} : x \in S\}| = 2$ pour tout $x \in U$. Il s'agit alors du PROBLÈME DE LA COUVERTURE PAR LES SOMMETS DE POIDS MINIMUM : étant donné un graphe G et $c : V(G) \to \mathbb{R}_+$, l'instance correspondante du problème de la couverture par des ensembles est définie par $U := E(G)$, $\mathcal{S} := \{\delta(v) : v \in V(G)\}$ et $c(\delta(v)) := c(v)$ pour tout $v \in V(G)$. Comme le PROBLÈME DE LA COUVERTURE PAR LES SOMMETS DE POIDS MINIMUM est *NP*-difficile même pour des poids unitaires (théorème 15.24), le PROBLÈME DE LA COUVERTURE MINIMUM PAR DES ENSEMBLES l'est également.

Johnson [1974] et Lovász [1975] ont proposé un simple algorithme glouton pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR DES ENSEMBLES : à chaque itération, choisir un ensemble qui couvre un nombre maximum d'éléments non encore couverts. Chvátal [1979] a généralisé cet algorithme au cas pondéré :

ALGORITHME GLOUTON DE LA COUVERTURE PAR DES ENSEMBLES

Input Un système d'ensembles $(U, \mathcal{S})$ tel que $\bigcup_{S\in\mathcal{S}} S = U$, des poids $c : \mathcal{S} \to \mathbb{R}_+$.

Output Une couverture par des ensembles $\mathcal{R}$ de $(U, \mathcal{S})$.

① Poser $\mathcal{R} := \emptyset$ et $W := \emptyset$.

② **While** $W \neq U$ **do** :
Choisir un ensemble $R \in \mathcal{S} \setminus \mathcal{R}$ tel que $\frac{c(R)}{|R\setminus W|}$ est minimum.
Poser $\mathcal{R} := \mathcal{R} \cup \{R\}$ et $W := W \cup R$.

Le temps de calcul est évidemment en $O(|U||\mathcal{S}|)$. On peut prouver la garantie de performance suivante :

Théorème 16.3. (Chvátal [1979]) *Pour toute instance $(U, \mathcal{S}, c)$ du* PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM, *l'*ALGORITHME GLOUTON DE LA COUVERTURE PAR DES ENSEMBLES *trouve une couverture par des ensembles dont le poids est au plus égal à $H(r)\,\mathrm{OPT}(U, \mathcal{S}, c)$, où $r := \max_{S\in\mathcal{S}} |S|$ et $H(r) = 1 + \frac{1}{2} + \cdots + \frac{1}{r}$.*

Preuve. Soit $(U, \mathcal{S}, c)$ une instance du PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM, et soit $\mathcal{R} = \{R_1, \ldots, R_k\}$ la solution trouvée par l'algorithme précédent, où R_i est l'ensemble choisi à la i-ième itération. Pour $j = 0, \ldots, k$ posons $W_j := \bigcup_{i=1}^{j} R_i$.

Pour chaque $e \in U$ soit $j(e) := \min\{j \in \{1, \ldots, k\} : e \in R_j\}$ l'itération où e est couvert. Posons

$$y(e) \;:=\; \frac{c(R_{j(e)})}{|R_{j(e)} \setminus W_{j(e)-1}|}.$$

Soit $S \in \mathcal{S}$ fixé, et soit $k' := \max\{j(e) : e \in S\}$. On a

$$\begin{aligned}
\sum_{e\in S} y(e) &= \sum_{i=1}^{k'} \sum_{e\in S: j(e)=i} y(e) \\
&= \sum_{i=1}^{k'} \frac{c(R_i)}{|R_i \setminus W_{i-1}|} |S \cap (W_i \setminus W_{i-1})| \\
&= \sum_{i=1}^{k'} \frac{c(R_i)}{|R_i \setminus W_{i-1}|} (|S \setminus W_{i-1}| - |S \setminus W_i|) \\
&\leq \sum_{i=1}^{k'} \frac{c(S)}{|S \setminus W_{i-1}|} (|S \setminus W_{i-1}| - |S \setminus W_i|)
\end{aligned}$$

d'après le choix de l'ensemble R_i à l'étape ② (observons que $S \setminus W_{i-1} \neq \emptyset$ pour $i = 1, \ldots, k'$). En écrivant $s_i := |S \setminus W_{i-1}|$, nous obtenons

$$\begin{aligned}
\sum_{e\in S} y(e) &\leq c(S)\sum_{i=1}^{k'} \frac{s_i - s_{i+1}}{s_i} \\
&\leq c(S)\sum_{i=1}^{k'} \left(\frac{1}{s_i} + \frac{1}{s_i - 1} + \cdots + \frac{1}{s_{i+1}+1}\right) \\
&= c(S)\sum_{i=1}^{k'} (H(s_i) - H(s_{i+1})) \\
&= c(S)(H(s_1) - H(s_{k'+1})) \\
&\leq c(S)H(s_1).
\end{aligned}$$

Comme $s_1 = |S| \leq r$, on conclut que

$$\sum_{e\in S} y(e) \;\leq\; c(S)H(r).$$

On somme sur tous les ensembles $S \in \mathcal{O}$ pour une couverture par des ensembles optimale $\mathcal{O}$ et on obtient

$$\begin{aligned}
c(\mathcal{O})H(r) &\geq \sum_{S\in\mathcal{O}} \sum_{e\in S} y(e) \\
&\geq \sum_{e\in U} y(e) \\
&= \sum_{i=1}^{k} \sum_{e\in U : j(e)=i} y(e) \\
&= \sum_{i=1}^{k} c(R_i) \;=\; c(\mathcal{R}).
\end{aligned}$$

□

Pour une analyse un peu plus fine du cas non pondéré, voir Slavík [1997]. Raz et Safra [1997] ont découvert qu'il existe une constante $c > 0$ telle que l'on ne puisse pas obtenir de rapport de performance égal à $c \ln |U|$, à moins que *P* = *NP*. En effet, un rapport de performance égal à $c \ln |U|$ ne peut pas être atteint quel que soit $c < 1$, à moins que chaque problème de *NP* puisse être résolu en temps $O\left(n^{O(\log\log n)}\right)$ (Feige [1998]).

Le PROBLÈME DE LA COUVERTURE PAR LES ARÊTES DE POIDS MINIMUM est évidemment un cas particulier du PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM. Nous avons ici $r = 2$ dans le théorème 16.3, ainsi l'algorithme précédent est un algorithme d'approximation de facteur $\frac{3}{2}$ dans ce cas particulier. Cependant, le problème peut aussi être résolu de manière optimale en temps polynomial ; voir l'exercice 11 du chapitre 11.

Pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS, l'algorithme précédent devient :

ALGORITHME GLOUTON DE LA COUVERTURE PAR LES SOMMETS	
Input	Un graphe G.
Output	Une couverture par les sommets R de G.

① Poser $R := \emptyset$.

② **While** $E(G) \neq \emptyset$ **do** :
Choisir un sommet $v \in V(G) \setminus R$ de degré maximum.
Poser $R := R \cup \{v\}$ et supprimer toutes les arêtes incidentes à v.

Cet algorithme semble raisonnable. On pourrait donc se demander pour quelle valeur de k il s'agirait d'un algorithme d'approximation de facteur k. Il peut paraître surprenant qu'une telle valeur n'existe pas. En effet, la borne donnée au théorème 16.3 est presque la meilleure possible :

Théorème 16.4. (Johnson [1974], Papadimitriou et Steiglitz [1982]) *Pour tout $n \geq 3$, il existe une instance G du* PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS *telle que $nH(n-1)+2 \leq |V(G)| \leq nH(n-1)+n$, le degré maximum de G est égal à $n-1$, $\mathrm{OPT}(G) = n$, et l'algorithme précédent peut trouver une couverture par les sommets les contenant tous sauf n d'entre eux.*

Preuve. Pour tout $n \geq 3$ et $i \leq n$, on définit $A_n^i := \sum_{j=2}^{i} \left\lfloor \frac{n}{j} \right\rfloor$ et

$$\begin{aligned} V(G_n) &:= \left\{a_1, \ldots, a_{A_n^{n-1}}, b_1, \ldots, b_n, c_1, \ldots, c_n\right\}. \\ E(G_n) &:= \{(b_i, c_i) : i = 1, \ldots, n\} \cup \\ &\qquad \bigcup_{i=2}^{n-1} \bigcup_{j=A_n^{i-1}+1}^{A_n^i} \left\{(a_j, b_k) : (j - A_n^{i-1} - 1)i + 1 \leq k \leq (j - A_n^{i-1})i\right\}. \end{aligned}$$

Observons que $|V(G_n)| = 2n + A_n^{n-1}$, $A_n^{n-1} \leq nH(n-1) - n$ et $A_n^{n-1} \geq nH(n-1) - n - (n-2)$. La figure 16.1 représente G_6.

Si nous appliquons notre algorithme à G_n, il peut d'abord choisir le sommet $a_{A_n^{n-1}}$ (car il est de degré maximum), puis les sommets $a_{A_n^{n-1}-1}, a_{A_n^{n-1}-2}, \ldots, a_1$. Il reste alors n arêtes disjointes qui ne sont incidentes à aucun de ces sommets. Donc n sommets supplémentaires sont nécessaires. Ainsi la couverture par les sommets construite est constituée de $A_n^{n-1} + n$ sommets, alors que la couverture par les sommets optimale $\{b_1, \ldots, b_n\}$ est de taille n. □

Il existe cependant des algorithmes d'approximation de facteur 2 pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS. Le plus simple est dû à Gavril (voir Garey et Johnson [1979]) : trouver un couplage maximum M et considérer l'ensemble formé des extrémités de toutes les arêtes de M. Il constitue évidemment une couverture par les sommets et contient $2|M|$ sommets. Puisque toute couverture par les sommets doit contenir $|M|$ sommets (aucun sommet ne couvre deux arêtes de M), il s'agit d'un algorithme d'approximation de facteur 2.

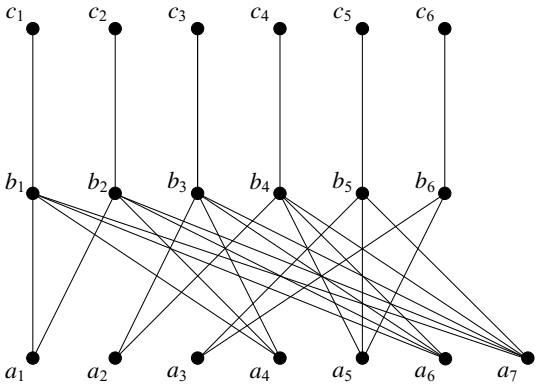


Figure 16.1.

Cette garantie de performance est forte : pensons simplement à un graphe constitué de nombreuses arêtes disjointes. Il peut paraître surprenant que l'algorithme précédent soit le meilleur algorithme d'approximation connu pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS. Nous montrerons plus tard qu'il existe un nombre $k > 1$ tel qu'aucun algorithme d'approximation de facteur k ne puisse exister à moins que *P = NP* (théorème 16.46). Plus précisément, il n'existe pas d'algorithme d'approximation de facteur $1,36$ à moins que *P = NP* (Dinur et Safra [2002]).

L'algorithme de Gavril peut être étendu au cas pondéré. Nous présentons l'algorithme de Bar-Yehuda et Even [1981], qui s'applique au PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM plus général :

ALGORITHME DE BAR-YEHUDA-EVEN

Input Un système d'ensembles $(U, \mathcal{S})$ tel que $\bigcup_{S\in\mathcal{S}} S = U$, des poids $c : \mathcal{S} \to \mathbb{R}_+$.

Output Une couverture par des ensembles $\mathcal{R}$ de $(U, \mathcal{S})$.

① Poser $\mathcal{R} := \emptyset$ et $W := \emptyset$. Poser $y(e) := 0$ pour tout $e \in U$.
Poser $c'(S) := c(S)$ pour tout $S \in \mathcal{S}$.

② **While** $W \neq U$ **do** :
Choisir un élément $e \in U \setminus W$.
Soit $R \in \mathcal{S}$ tel que $e \in R$ et que $c'(R)$ soit minimum.
Poser $y(e) := c'(R)$.
Poser $c'(S) := c'(S) - y(e)$ pour tout $S \in \mathcal{S}$ tel que $e \in S$.
Poser $\mathcal{R} := \mathcal{R} \cup \{R\}$ et $W := W \cup R$.

Théorème 16.5. (Bar-Yehuda et Even [1981]) *Pour toute instance $(U, \mathcal{S}, c)$ du* PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM, *l'*ALGORITHME DE BAR-YEHUDA-EVEN *trouve une couverture par des ensembles*

dont le poids est au plus égal à $p\,\mathrm{OPT}(U,\mathcal{S},c)$*, où* $p := \max_{e\in U} |\{S \in \mathcal{S} : e \in S\}|$.

Preuve. Le PROBLÈME DE LA COUVERTURE PAR DES ENSEMBLES DE POIDS MINIMUM peut être écrit sous la forme du PL en nombres entiers suivant

$$\min\left\{cx : Ax \geq \mathbb{1},\ x \in \{0,1\}^{\mathcal{S}}\right\},$$

où A est la matrice dont les lignes correspondent aux éléments de U et dont les colonnes correspondent aux vecteurs d'incidence des ensembles de $\mathcal{S}$. L'optimum de la relaxation linéaire

$$\min\{cx : Ax \geq \mathbb{1},\ x \geq 0\}$$

donne une borne inférieure pour $\mathrm{OPT}(U,\mathcal{S},c)$ (l'omission des contraintes $x \leq \mathbb{1}$ ne modifie pas la valeur optimale de ce PL). Ainsi, d'après la proposition 3.13, l'optimum du PL dual

$$\max\{y\mathbb{1} : yA \leq c,\ y \geq 0\}$$

est aussi une borne inférieure pour $\mathrm{OPT}(U,\mathcal{S},c)$.

Observons alors que $c'(S) \geq 0$ pour tout $S \in \mathcal{S}$ à n'importe quelle étape de l'algorithme. Ainsi $y \geq 0$ et $\sum_{e\in S} y(e) \leq c(S)$ pour tout $S \in \mathcal{S}$, c.-à-d. y est une solution réalisable du PL dual et

$$y\mathbb{1} \ \leq\ \max\{y\mathbb{1} : yA \leq c,\ y \geq 0\} \ \leq\ \mathrm{OPT}(U,\mathcal{S},c).$$

Finalement observons que

$$\begin{aligned} c(\mathcal{R}) &= \sum_{R\in\mathcal{R}} c(R) \\ &= \sum_{R\in\mathcal{R}} \sum_{e\in R} y(e) \\ &\leq \sum_{e\in U} py(e) \\ &= py\mathbb{1} \\ &\leq p\,\mathrm{OPT}(U,\mathcal{S},c). \end{aligned}$$

□

Puisque nous avons $p = 2$ dans le cas de la couverture par les sommets, c'est un algorithme d'approximation de facteur 2 pour le PROBLÈME DE LA COUVERTURE PAR LES SOMMETS DE POIDS MINIMUM. Le premier algorithme d'approximation de facteur 2 est dû à Hochbaum [1982]. Elle proposa de chercher une solution optimale y du PL dual dans la preuve précédente et de prendre tous les ensembles S tels que $\sum_{e\in S} y(e) = c(S)$. Alternativement, on pourrait chercher une solution optimale x du PL primal et prendre tous les ensembles S tels que $x_S \geq \frac{1}{p}$.

L'avantage de l'ALGORITHME DE BAR-YEHUDA-EVEN est qu'il n'utilise pas explicitement la programmation linéaire. En fait, il peut facilement être implémenté de telle manière qu'il s'exécute en temps $O\left(\sum_{S\in\mathcal{S}} |S|\right)$. C'est notre premier exemple d'un algorithme primal-dual d'approximation. Nous verrons des exemples plus complexes aux paragraphes 20.4 et 22.3.

16.2 Problème de la coupe-max

Dans ce paragraphe, nous considérons un autre problème de base :

PROBLÈME DE LA COUPE DE POIDS MAXIMUM	
Instance	Un graphe non orienté G et des poids $c : E(G) \to \mathbb{Z}_+$.
Tâche:	Trouver une coupe dans G de poids total maximum.

Ce problème est souvent appelé plus simplement COUPE-MAX. Contrairement au problème de la coupe de capacité minimum, présenté au paragraphe 8.7, cela est un problème difficile. Il est fortement *NP*-difficile ; même le cas particulier où $c \equiv 1$ (le PROBLÈME DE LA COUPE MAXIMUM) est difficile :

Théorème 16.6. (Garey, Johnson et Stockmeyer [1976]) *Le* PROBLÈME DE LA COUPE MAXIMUM *est NP-difficile.*

Preuve. Par réduction à partir du problème MAX-2SAT (voir théorème 15.36). Étant donné une instance de MAX-2SAT avec n variables et m clauses, on construit un graphe G dont les sommets correspondent aux littéraux plus un sommet supplémentaire z. Pour chaque variable x, on ajoute $3m$ arêtes parallèles entre x et $\bar{x}$. Pour chaque clause $\{\lambda, \lambda'\}$, on ajoute trois arêtes (λ, λ'), (λ, z) et (λ', z). G a donc $2n + 1$ sommets et $3m(n + 1)$ arêtes.

Nous affirmons que la cardinalité maximum d'une coupe de G est égale à $3mn + 2t$, où t est le nombre maximum de clauses satisfaites par un assignement. En effet, étant donné un assignement satisfaisant t clauses, considérons l'ensemble X des littéraux vrais. Alors $|\delta_G(X)| = 3mn + 2t$. Inversement, s'il existe un ensemble $X \subseteq V(G)$ tel que $|\delta_G(X)| \geq 3mn + a$ alors, sans perte de généralité, $z \notin X$ (sinon remplacer X par $V(G) \setminus X$), et pour chaque variable x on a $|X \cap \{x, \bar{x}\}| = 1$ (sinon remplacer X par $X \Delta \{x\}$ et augmenter la coupe). Ainsi on peut mettre tous les littéraux de X à *vrai* et obtenir un assignement satisfaisant au moins $\frac{a}{2}$ clauses. □

Il est facile de trouver un algorithme d'approximation de facteur 2 pour le PROBLÈME DE LA COUPE DE POIDS MAXIMUM :
Si $V(G) = \{v_1, \ldots, v_n\}$, commencer avec $X := \{v_1\}$, et pour $i = 3, \ldots, n$ ajouter v_i à X si $\sum_{e\in E(v_i,\{v_1,\ldots,v_{i-1}\}\cap X)} c(e) < \sum_{e\in E(v_i,\{v_1,\ldots,v_{i-1}\}\setminus X)} c(e)$. (L'analyse de cet algorithme est simple et est renvoyée à l'exercice 9.)

Pendant longtemps ce fut le seul algorithme d'approximation connu. Ensuite Goemans et Williamson [1995] trouvèrent un bien meilleur algorithme utilisant la programmation semi-définie. Le reste de ce paragraphe est fondé sur leur article.

Soit G un graphe non orienté et $c : E(G) \to \mathbb{R}_+$. Sans perte de généralité, $V(G) = \{1, \ldots, n\}$. Pour $1 \leq i, j \leq n$ notons $c_{ij} := c((i,j))$ si $(i,j) \in E(G)$ et $c_{ij} := 0$ sinon. Alors le PROBLÈME DE LA COUPE DE POIDS MAXIMUM consiste à trouver un sous-ensemble $S \subseteq \{1, \ldots, n\}$ maximisant $\sum_{i \in S,\, j \in \{1,\ldots,n\} \setminus S} c_{ij}$. En remplaçant S par $y \in \{-1, 1\}^n$ avec $y_i = 1$ si et seulement si $i \in S$, nous pouvons formuler le problème de la manière suivante :

$$\begin{array}{lll} \max & \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - y_i y_j) & \\ \text{s.c.} & y_i \in \{-1, 1\} & (i = 1, \ldots, n). \end{array}$$

Les variables y_i peuvent être considérées comme des vecteurs de dimension 1 et de norme unitaire. En les considérant comme des vecteurs multidimensionnels de norme euclidienne unitaire, nous obtenons une relaxation très intéressante :

$$\begin{array}{lll} \max & \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - y_i^\top y_j) & \\ \text{s.c.} & y_i \in \mathcal{S}_{m-1} & (i = 1, \ldots, n) \end{array} \tag{16.1}$$

où $m := |E(G)|$ et $\mathcal{S}_{m-1} = \{x \in \mathbb{R}^m : ||x||_2 = 1\}$ désigne la sphère unitaire de dimension $(m-1)$. Par exemple, pour le triangle ($n = 3$, $c_{12} = c_{13} = c_{23} = 1$) l'optimum est obtenu pour des points, situés sur la sphère unité de dimension 2, qui sont les sommets d'un triangle équilatéral, par exemple, $y_1 = (0, -1)$, $y_2 = (-\frac{\sqrt{3}}{2}, \frac{1}{2})$, et $y_3 = (\frac{\sqrt{3}}{2}, \frac{1}{2})$, qui donnent une valeur optimale égale à $\frac{9}{4}$, alors que le poids maximum d'une coupe est égal à 2. Cependant, l'intérêt est que nous pouvons résoudre (16.1) presque optimalement en temps polynomial.

L'idée est de ne pas considérer directement les variables y_i, pas même leur dimension. Au lieu de cela, nous considérons la matrice $n \times n$ $(y_i^\top y_j)_{i,j=1,\ldots,n}$. Puisqu'une matrice X est semi-définie positive si et seulement si elle peut être écrite sous la forme $B^\top B$ pour une certaine matrice B, on peut écrire de manière équivalente

$$\begin{array}{lll} \max & \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - x_{ij}) & \\ \text{s.c.} & x_{ii} = 1 & (i = 1, \ldots, n) \\ & X = (x_{ij})_{1 \leq i,j \leq n} & \text{symétrique et semi-définie positive.} \end{array} \tag{16.2}$$

À partir d'une solution de (16.2), nous pouvons obtenir une solution de (16.1) avec $m \leq n$ par une factorisation de Cholesky en temps $O(n^3)$ (nous devons accepter une erreur d'arrondi arbitrairement petite ; voir exercice 6 du chapitre 4).

Le problème (16.2) est appelé un programme semi-défini. Il peut être résolu approximativement en temps polynomial par la MÉTHODE DES ELLIPSOÏDES, en appliquant le théorème 4.19, comme nous allons le voir maintenant. Observons d'abord que nous optimisons une fonction objectif linéaire sur l'ensemble convexe

$$P := \big\{X = (x_{ij})_{1\leq i,j\leq n} \in \mathbb{R}^{n\times n} : X \text{ symétrique et semi-définie positive,} \\ x_{ii} = 1\,(i = 1,\ldots,n)\big\}.$$

En projetant P par rapport aux $\frac{n^2-n}{2}$ variables libres, on obtient

$$P' := \{(x_{ij})_{1\leq i<j\leq n} : (x_{ij})_{1\leq i,j\leq n} \in P, \text{ où } x_{ii} := 1 \text{ et } x_{ji} := x_{ij} \text{ pour } i < j\}.$$

Remarquons que P et P' ne sont pas des polyèdres. Cependant, P' est convexe, borné, et de pleine dimension :

Proposition 16.7. *P' est convexe. De plus, $B(0, \frac{1}{n}) \subseteq P' \subseteq B(0, n)$.*

Preuve. La convexité vient du simple fait que des combinaisons convexes de matrices semi-définies positives sont semi-définies positives.

Pour la première inclusion, observons que pour une matrice symétrique $n \times n$ X dont les termes diagonaux sont égaux à 1 et dont les autres termes ont leurs valeurs absolues au plus égales à $\frac{1}{n}$ on a, pour tout $d \in \mathbb{R}^n$, $d^\top X d = \sum_{i,j=1}^n x_{ij} d_i d_j \geq \frac{1}{2n-2} \sum_{i\neq j}(x_{ii}d_i^2 + x_{jj}d_j^2 - (2n-2)|x_{ij}||d_i d_j|) \geq \frac{1}{2n-2}\sum_{i\neq j}^n (d_i^2 + d_j^2 - 2|d_i d_j|) = \frac{1}{2n-2}\sum_{i\neq j}^n (|d_i| - |d_j|)^2 \geq 0$, c.-à-d. X est semi-définie positive.

Pour la seconde inclusion, remarquons que tous les termes non diagonaux d'une matrice dans P ont leurs valeurs absolues au plus égales à 1, et ainsi la norme euclidienne du vecteur composé des termes au-dessus de la diagonale est au plus égale à n. □

Il reste à montrer que le PROBLÈME DE SÉPARATION pour P' peut être résolu en temps polynomial. Cela est réalisé par ÉLIMINATION GAUSSIENNE :

Théorème 16.8. *Étant donné une matrice symétrique $X \in \mathbb{Q}^{n\times n}$, on peut décider en temps polynomial si X est semi-définie positive, et trouver un vecteur $d \in \mathbb{Q}^n$ tel que $d^\top X d < 0$ s'il en existe un.*

Preuve. Si $x_{nn} < 0$, alors on pose $d = (0,\ldots,0,1)$ et on a $d^\top X d < 0$. Si $x_{nn} = 0$ et $x_{nj} \neq 0$ pour un certain $j < n$, alors on peut définir d par $d_j := -1$, $d_n := \frac{x_{jj}}{2x_{nj}} + x_{nj}$, et $d_i := 0$ pour $i \in \{1,\ldots,n-1\} \setminus \{j\}$, et on obtient $d^\top X d = x_{jj} - 2x_{nj}(\frac{x_{jj}}{2x_{nj}} + x_{nj}) = -2(x_{nj})^2 < 0$, ce qui prouve encore que X n'est pas semi-définie positive.

Dans les autres cas on réduit la dimension. Si $x_{nj} = 0$ pour tout j, alors la dernière ligne et la dernière colonne peuvent être supprimées : X est semi-définie

positive si et seulement si $X' := (x_{ij})_{i,j=1,\dots,n-1}$ est semi-définie positive. De plus, si $c \in \mathbb{Q}^{n-1}$ vérifie $c^\top X' c < 0$, on pose $d := (c^\top, 0)^\top$ et on obtient $d^\top X d < 0$.

Si $x_{nn} > 0$, posons $X' := (x_{ij} - \frac{x_{ni}x_{nj}}{x_{nn}})_{i,j=1,\dots,n-1}$; cela correspond à une itération de l'ÉLIMINATION GAUSSIENNE. Remarquons que X' est semi-définie positive si et seulement si X est semi-définie positive.

Si $c \in \mathbb{Q}^{n-1}$ est tel que $c^\top X' c < 0$, on posera $d := (c^\top, -\frac{1}{x_{nn}} \sum_{i=1}^{n-1} c_i x_{ni})^\top$. Alors

$$\begin{aligned} d^\top X d &= \sum_{i,j=1}^{n-1} d_i \left(x'_{ij} + \frac{x_{ni}}{x_{nn}} x_{nj} \right) d_j + 2 \sum_{j=1}^{n-1} d_n x_{nj} d_j + d_n^2 x_{nn} \\ &= c^\top X' c + \sum_{i,j=1}^{n-1} c_i \frac{x_{ni} x_{nj}}{x_{nn}} c_j (1 - 2 + 1) \\ &= c^\top X' c \\ &< 0. \end{aligned}$$

Cela définit un algorithme polynomial. Afin de voir que les nombres mis en jeu dans le calcul de d ne sont pas trop grands, notons $X^{(n)}, X^{(n-1)}, \dots, X^{(k)}$ les matrices considérées ($X^{(i)} \in \mathbb{Q}^{i \times i}$) et supposons que nous observons à l'itération $n+1-k$ que la matrice $X^{(k)} = (y_{ij})_{i,j=1,\dots,k}$ n'est pas semi-définie positive (c.-à-d. $y_{kk} < 0$ ou $y_{kk} = 0$ et $y_{kj} \neq 0$ pour un certain $j < k$). On a un vecteur $c \in \mathbb{Q}^k$ tel que $c^\top X^{(k)} c < 0$ et $\text{taille}(c) \leq 2\,\text{taille}(X^{(k)})$. On peut maintenant construire un vecteur $d \in \mathbb{Q}^n$ tel que $d^\top X d < 0$ comme ci-dessus. Remarquons que d est une solution du système d'équations linéaire $Md = (c^\top, 0)^\top$, où la j-ième ligne de M est :

- le j-ième vecteur unitaire si $j \leq k$;
- le j-ième vecteur unitaire si $j > k$ et que la j-ième ligne de $X^{(j)}$ ne contient que des zéros ;
- la j-ième ligne de $X^{(j)}$, suivie de zéros, sinon.

Ainsi, d'après le théorème 4.4, on a $\text{taille}(d) \leq 4n(\text{taille}(M) + \text{taille}(c))$, ce qui est polynomial d'après le théorème 4.10. □

Corollaire 16.9. *Le* PROBLÈME DE SÉPARATION *pour P' peut être résolu en temps polynomial.*

Preuve. Étant donné $(y_{ij})_{1 \leq i < j \leq n}$, et soit $Y = (y_{ij})_{1 \leq i,j \leq n}$ la matrice symétrique définie par $y_{ii} = 1$ pour tout i et $y_{ji} := y_{ij}$ pour $i < j$. Appliquons le théorème 16.8. Si Y est semi-définie positive, nous avons fini.

Sinon on recherche un vecteur $d \in \mathbb{Q}^n$ tel que $d^\top Y d < 0$. Alors $-\sum_{i=1}^n d_i^2 > d^\top Y d - \sum_{i=1}^n d_i^2 = \sum_{1 \leq i < j \leq n} 2 d_i d_j y_{ij}$, mais $\sum_{1 \leq i < j \leq n} 2 d_i d_j z_{ij} \geq -\sum_{i=1}^n d_i^2$ pour tout $z \in P'$. Ainsi $(d_i d_j)_{1 \leq i < j \leq n}$ constitue un hyperplan séparateur. □

On peut maintenant conclure :

Théorème 16.10. *Pour toute instance du* PROBLÈME DE LA COUPE DE POIDS MAXIMUM*, on peut trouver un $Y \in P$ tel que*

$$\sum_{1\leq i<j\leq n} c_{ij}(1-y_{ij}) \geq \max\left\{\sum_{1\leq i<j\leq n} c_{ij}(1-x_{ij}) : X \in P\right\} - \epsilon$$

en temps polynomial par rapport à n, $\text{taille}((c_{ij})_{1\leq i<j\leq n})$, *et* $\text{taille}(\epsilon)$.

Preuve. On applique le théorème 4.19, en utilisant la proposition 16.7 et le corollaire 16.9. □

Les programmes semi-définis tels que (16.2) peuvent aussi être résolus approximativement à l'aide d'algorithmes de points intérieurs, qui sont plus efficaces que la MÉTHODE DES ELLIPSOÏDES. Voir Alizadeh [1995] pour plus de détails.

Comme il est mentionné précédemment, on peut, à partir d'une solution presque optimale de (16.2), obtenir une solution de (16.1) avec la même valeur de fonction objectif par factorisation de Cholesky. Cette solution est constituée d'un ensemble de vecteurs $y_i \in \mathbb{R}^m$ $(i = 1, \ldots, n)$ pour un certain $m \leq n$. Comme (16.1) est une relaxation de notre problème de départ, on a que l'optimum est au plus égal à $\frac{1}{2}\sum_{1\leq i<j\leq n} c_{ij}(1 - y_i^\top y_j) + \epsilon$.

Les vecteurs y_i sont situés sur la sphère unité de dimension $(m-1)$. L'idée est maintenant de choisir aléatoirement un hyperplan passant par l'origine, et de définir S comme l'ensemble des indices i pour lesquels y_i est d'un côté de cet hyperplan.

Un hyperplan aléatoire passant par l'origine correspond à un point aléatoire sur la sphère unité de dimension$(m-1)$. Celui-ci peut être déterminé en tirant indépendamment m nombres réels selon une loi de distribution normale, ce qui à son tour peut être réalisé en utilisant des nombres aléatoires indépendants uniformément distribués sur l'intervalle $[0, 1]$. Voir Knuth [1969] (paragraphe 3.4.1) pour plus de détails.

L'algorithme de Goemans et Williamson peut maintenant être décrit de la manière suivante.

ALGORITHME DE GOEMANS-WILLIAMSON POUR LA COUPE-MAX

Input Un nombre $n \in \mathbb{N}$, des nombres $c_{ij} \geq 0$ pour $1 \leq i < j \leq n$.

Output Un ensemble $S \subseteq \{1, \ldots, n\}$.

① Résoudre (16.2) approximativement, c.-à-d. trouver une matrice semi-définie positive symétrique $X = (x_{ij})_{1\leq i,j\leq n}$ avec $x_{ii} = 1$ pour $i = 1, \ldots, n$, telle que $\sum_{1\leq i<j\leq n} c_{ij}(1 - x_{ij})$ soit au moins égal à 0.9995 fois la valeur maximale possible.

② Appliquer la factorisation de Cholesky à X afin d'obtenir des vecteurs $y_1, \ldots, y_n \in \mathbb{R}^m$ tels que $m \leq n$ et $y_i^\top y_j = x_{ij}$ pour tout $i, j \in \{1, \ldots, n\}$.

③ Choisir un point aléatoire a sur la sphère unité de dimension $(m-1)$ $\{x \in \mathbb{R}^m : ||x||_2 = 1\}$.

④ Poser $S := \{i \in \{1, \ldots, n\} : a^\top y_i \geq 0\}$.

Théorème 16.11. *L'*ALGORITHME DE GOEMANS-WILLIAMSON POUR LA COUPE-MAX *s'exécute en temps polynomial.*

Preuve. Voir la discussion précédente. L'étape la plus difficile, ①, peut être réalisée en temps polynomial d'après le théorème 16.10. On peut ici choisir $\epsilon = 0,00025 \sum_{1\le i<j\le n} c_{ij}$, car $\frac{1}{2}\sum_{1\le i<j\le n} c_{ij}$ est une borne inférieure de la valeur de la fonction objectif (atteinte en choisissant aléatoirement $S \subseteq \{1,\dots,n\}$) et donc de la valeur optimale de (16.2). □

Nous pouvons donner maintenant la garantie de performance :

Théorème 16.12. (Goemans et Williamson [1995]) *L'*ALGORITHME DE GOEMANS-WILLIAMSON POUR LA COUPE-MAX *fournit un ensemble S pour lequel la valeur espérée de $\sum_{i\in S,\, j\notin S} c_{ij}$ est au moins égale à $0,878$ fois la valeur maximale possible.*

Preuve. Notons de nouveau $\mathcal{S}_{m-1}$ la sphère unité de dimension $(m-1)$, et soit $H(y) := \{x \in \mathcal{S}_{m-1} : x^\top y \ge 0\}$ l'hémisphère de pôle y, pour $y \in \mathcal{S}_{m-1}$. Pour un sous-ensemble $A \subseteq \mathcal{S}_{m-1}$, notons $\mu(A) = \frac{\text{volume}\,(A)}{\text{volume}\,(\mathcal{S}_{m-1})}$; cela définit une mesure de probabilité sur $\mathcal{S}_{m-1}$. On a $|S \cap \{i,j\}| = 1$ avec probabilité $\mu(H(y_i)\triangle H(y_j))$, où $\triangle$ désigne la différence symétrique. Remarquons que $H(y_i)\triangle H(y_j)$ est l'union de deux digones sphériques, chacun ayant un angle égal à $\arccos(y_i^\top y_j)$. Le volume étant proportionnel à l'angle, on a $\mu(H(y_i)\triangle H(y_j)) = \frac{1}{\pi}\arccos(y_i^\top y_j)$.

Affirmation : $\frac{1}{\pi}\arccos\beta \ge 0,8785\frac{1-\beta}{2}$ pour tout $\beta \in [-1,1]$.

Pour $\beta = 1$ on a égalité. De plus, un calcul élémentaire donne

$$\min_{-1\le\beta<1} \frac{\arccos\beta}{1-\beta} = \min_{0<\gamma\le\pi} \frac{\gamma}{1-\cos\gamma} = \frac{1}{\sin\gamma'},$$

où γ' est déterminé par $\cos\gamma' + \gamma'\sin\gamma' = 1$. On obtient $2,3311 < \gamma' < 2,3312$ et $\frac{1}{\sin\gamma'} > \frac{1}{\sin 2,3311} > 1,38$. Comme $\frac{1,38}{\pi} > \frac{0,8785}{2}$, l'affirmation est prouvée.

Ainsi la valeur espérée de $\sum_{i\in S,\, j\notin S} c_{ij}$ est

$$\begin{aligned}
\sum_{1\le i<j\le n} c_{ij}\mu(H(y_i)\triangle H(y_j)) &= \sum_{1\le i<j\le n} c_{ij}\frac{1}{\pi}\arccos(y_i^\top y_j)\\
&\ge 0,8785\cdot\frac{1}{2}\sum_{1\le i<j\le n} c_{ij}(1-y_i^\top y_j)\\
&= 0,8785\cdot\frac{1}{2}\sum_{1\le i<j\le n} c_{ij}(1-x_{ij})\\
&\ge 0,8785\cdot 0,9995\cdot \mathrm{OPT}(16.2)\\
&> 0,878\cdot\mathrm{OPT}(16.2)\\
&\ge 0,878\cdot\max\left\{\sum_{i\in S,\, j\notin S} c_{ij} : S\subseteq\{1,\dots,n\}\right\}.
\end{aligned}$$

□

Ainsi l'algorithme est appelé un algorithme d'approximation randomisé de facteur $1,139$. Mahajan et Ramesh [1999] ont montré comment dérandomiser cet algorithme. On obtient ainsi un algorithme d'approximation déterministe de facteur $1,139$. Cependant, il n'existe pas d'algorithme d'approximation de facteur $1,062$ à moins que $P = NP$ (Håstad [2001], Papadimitriou et Yannakakis [1991]).

Voir Lovász [2003] pour d'autres liens intéressants entre la programmation semi-définie et l'optimisation combinatoire.

16.3 Coloration

Dans ce paragraphe, nous présentons brièvement deux cas particuliers supplémentaires du PROBLÈME DE LA COUVERTURE MINIMUM PAR DES ENSEMBLES : nous voulons partitionner l'ensemble des sommets d'un graphe en ensembles stables, ou l'ensemble des arêtes d'un graphe en couplages :

Définition 16.13. *Soit G un graphe non orienté. Une* **coloration des sommets** *de G est une application $f : V(G) \to \mathbb{N}$ telle que $f(v) \neq f(w)$ pour tout $\{v, w\} \in E(G)$. Une* **coloration des arêtes** *de G est une application $f : E(G) \to \mathbb{N}$ telle que $f(e) \neq f(e')$ pour tout $e, e' \in E(G)$ avec $e \neq e'$ et $e \cap e' \neq \emptyset$.*

Le nombre $f(v)$ ou $f(e)$ est appelé la couleur de v ou e. En d'autres termes, l'ensemble des sommets ou des arêtes de la même couleur (f-valeur) doit être, respectivement, un ensemble stable, ou un couplage. Bien entendu notre but est d'utiliser le nombre minimum de couleurs :

PROBLÈME DE LA COLORATION DES SOMMETS	
Instance	Un graphe non orienté G.
Tâche	Trouver une coloration des sommets $f : V(G) \to \{1, \ldots, k\}$ de G avec k minimum.

PROBLÈME DE LA COLORATION DES ARÊTES	
Instance	Un graphe non orienté G.
Tâche	Trouver une coloration des arêtes $f : E(G) \to \{1, \ldots, k\}$ de G avec k minimum.

Réduire ces problèmes au PROBLÈME DE LA COUVERTURE MINIMUM PAR DES ENSEMBLES n'est pas très utile : pour le PROBLÈME DE LA COLORATION DE SOMMETS nous aurions à établir la liste des ensembles stables maximaux (un problème *NP*-difficile), alors que pour le PROBLÈME DE LA COLORATION DES ARÊTES nous aurions à considérer un nombre exponentiel de couplages maximaux.

La valeur optimale du PROBLÈME DE LA COLORATION DE SOMMETS (c.-à-d. le nombre minimum de couleurs) est appelée le **nombre chromatique** du graphe.

La valeur optimale du PROBLÈME DE LA COLORATION DES ARÊTES est appelée le **nombre arête-chromatique** ou parfois l'indice chromatique. Ces deux problèmes de coloration sont *NP*-difficiles :

Théorème 16.14. *Les problèmes de décision suivants sont NP-complets :*
(a) (Holyer [1981]) *Décider si un graphe simple donné a son nombre arête-chromatique égal à 3.*
(b) (Stockmeyer [1973]) *Décider si un graphe planaire donné a son nombre chromatique égal à 3.*

Les problèmes restent *NP*-difficiles même si le graphe a son degré maximum égal à trois dans (a), et égal à quatre dans (b).

Proposition 16.15. *Pour tout graphe donné, on peut décider en temps linéaire si son nombre chromatique est inférieur à 3 et, si c'est le cas, trouver une coloration optimale. Le même résultat est vérifié par le nombre arête-chromatique.*

Preuve. Un graphe a son nombre chromatique égal à 1 si et seulement s'il n'a pas d'arêtes. Par définition, les graphes de nombre chromatique au plus égal à 2 sont précisément les graphes bipartis. D'après la proposition 2.27, on peut vérifier en temps linéaire si un graphe est biparti et, si c'est le cas, trouver une bipartition, c.-à-d. une coloration des sommets avec deux couleurs.

Pour vérifier si le nombre arête-chromatique d'un graphe G est inférieur à 3 (et, si c'est le cas, trouver une coloration optimale des arêtes) on considère simplement le PROBLÈME DE LA COLORATION DES SOMMETS appliqué au *line graph* de G. Cela est évidemment équivalent. □

Pour les graphes bipartis, le PROBLÈME DE LA COLORATION DES ARÊTES peut également être résolu :

Théorème 16.16. (König [1916]) *Le nombre arête-chromatique d'un graphe biparti G est égal au degré maximum des sommets de G.*

Preuve. Par induction sur $|E(G)|$. Soit G un graphe de degré maximum égal à k, et soit $e = (v, w)$ une arête. Par l'hypothèse d'induction, $G - e$ possède une coloration des arêtes f avec k couleurs. Il y a des couleurs $i, j \in \{1, \ldots, k\}$ telles que $f(e') \neq i$ pour tout $e' \in \delta(v)$ et $f(e') \neq j$ pour tout $e' \in \delta(w)$. Si $i = j$, nous avons terminé puisque nous pouvons étendre f à G en donnant à e la couleur i.

Le graphe $H = (V(G), \{e' \in E(G) \setminus e : f(e') \in \{i, j\}\})$ a son degré maximum égal à 2, et v est de degré au plus 1 dans H. Soit P la chaîne maximum de H ayant v comme extrémité. Les couleurs alternent sur P ; ainsi l'autre extrémité de P ne peut pas être w. En échangeant les couleurs i et j sur P on étend la coloration des arêtes à G en donnant à e la couleur j. □

Le degré maximum d'un sommet fournit une borne inférieure évidente du nombre arête-chromatique d'un graphe. Cette borne n'est pas toujours atteinte comme le montre le triangle K_3. Le théorème suivant montre comment trouver une coloration des arêtes, d'un graphe simple donné, qui nécessite au plus une couleur de plus que nécessaire :

Théorème 16.17. (Vizing [1964]) *Soit G un graphe simple non orienté de degré maximum égal à k. Alors G a une coloration des arêtes à l'aide d'au plus $k+1$ couleurs, et une telle coloration peut être trouvée en temps polynomial.*

Preuve. Par induction sur $|E(G)|$. Si G n'a pas d'arêtes, la proposition est évidente. Sinon, considérons une arête $e = (x, y_0)$. Par l'hypothèse d'induction, il existe une coloration des arêtes f de $G-e$ à l'aide de $k+1$ couleurs. Pour chaque sommet v choisissons une couleur $n(v) \in \{1, \ldots, k+1\} \setminus \{f(w) : w \in \delta_{G-e}(v)\}$ non utilisée en v.

En partant de y_0, construisons une suite maximale $y_0, y_1, \ldots, y_t$ de voisins distincts de x tels que $n(y_{i-1}) = f((x, y_i))$ pour $i = 1, \ldots, t$.

Si aucune arête incidente à x n'est de couleur $n(y_t)$, alors on construit une coloration des arêtes f' de G à partir de f en posant $f'((x, y_{i-1})) := f((x, y_i))$ $(i = 1, \ldots, t)$ et $f'((x, y_t)) := n(y_t)$. Supposons donc qu'il existe une arête incidente à x de couleur $n(y_t)$; par la maximalité de t, on a $f((x, y_s)) = n(y_t)$ pour un certain $s \in \{1, \ldots, t-1\}$.

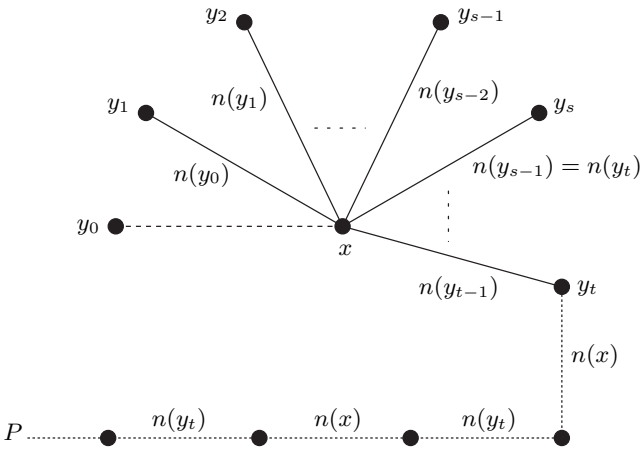


Figure 16.2.

Considérons la chaîne maximum P partant de y_t dans le graphe $(V(G), \{e' \in E(G-e) : f(e') \in \{n(x), n(y_t)\}\})$ (ce graphe est de degré maximum égal à 2 ; voir figure 16.2). Nous distinguons deux cas. Dans chaque cas, nous construisons une coloration des arêtes f' de G.

Si P se termine en x, alors (y_s, x) est la dernière arête de P. On construit f' à partir de f en échangeant les couleurs $n(x)$ et $n(y_t)$ sur P, et en posant $f'((x, y_{i-1})) := f((x, y_i))$ $(i = 1, \ldots, s)$.

Si P se termine en y_{s-1}, alors la dernière arête de P a la couleur $n(x)$, puisque la couleur $n(y_t) = f((x, y_s)) = n(y_{s-1})$ n'est pas utilisée en y_{s-1}. On construit f' à partir de f de la manière suivante : échangeons les couleurs $n(x)$ et $n(y_t)$ sur P, posons $f'((x, y_{i-1})) := f((x, y_i))$ $(i = 1, \ldots, s-1)$ et $f'((x, y_{s-1})) := n(x)$.

Si P ne se termine ni en x ni en y_{s-1}, alors on peut construire f' à partir de f en échangeant les couleurs $n(x)$ et $n(y_t)$ sur P, et en posant $f'((x, y_{i-1})) := f((x, y_i))$ $(i = 1, \ldots, t)$ et $f'((x, y_t)) := n(x)$. □

Le théorème de Vizing implique un algorithme d'approximation absolue pour le PROBLÈME DE LA COLORATION DES ARÊTES dans les graphes simples. Si on autorise les arêtes parallèles, l'affirmation du théorème de Vizing n'est plus vraie : en remplaçant chaque arête du triangle K_3 par r arêtes parallèles, on obtient un graphe $2r$-régulier de nombre arête-chromatique égal à $3r$.

Nous passons maintenant au PROBLÈME DE LA COLORATION DES SOMMETS. Le degré maximum fournit également une borne supérieure du nombre chromatique :

Théorème 16.18. *Soit G un graphe de degré maximum égal à k. Alors G a une coloration des sommets à l'aide d'au plus $k + 1$ couleurs, et une telle coloration peut être trouvée en temps linéaire.*

Preuve. L'ALGORITHME GLOUTON DE COLORATION suivant trouve de façon évidente une telle coloration. □

ALGORITHME GLOUTON DE COLORATION	
Input	Un graphe non orienté G.
Output	Une coloration des sommets de G.

① Soit $V(G) = \{v_1, \ldots, v_n\}$.

② **For** $i := 1$ **to** n **do** :
Poser $f(v_i) := \min\{k \in \mathbb{N} : k \neq f(v_j)$ pour tout $j < i$ tel que $v_j \in \Gamma(v_i)\}$.

Pour les graphes complets et pour les cycless impairs, on a évidemment besoin de $k+1$ couleurs, où k est le degré maximum. Pour tous les autres graphes connexes k couleurs suffisent, comme l'a montré Brooks [1941]. Cependant, le degré maximum ne constitue pas une borne inférieure du nombre chromatique : toute étoile $K_{1,n}$ $(n \in \mathbb{N})$ a un nombre chromatique égal à 2. Ainsi ces résultats ne mènent pas à un algorithme d'approximation. En fait, on ne connaît aucun algorithme pour le PROBLÈME DE LA COLORATION DES SOMMETS qui ait une garantie de performance raisonnable pour les graphes généraux ; voir Khanna, Linial et Safra [2000]. Zuckerman [2006] a montré qu'il n'existe aucun algorithme polynomial calculant le nombre chromatique d'un graphe sur n sommets à un facteur $n^{1-\epsilon}$ près, pour tout $\epsilon > 0$ fixé, à moins que *P = NP*.

Puisque le degré maximum n'est pas une borne inférieure pour le nombre chromatique, on peut considérer la taille maximum d'une clique. Évidemment, si un graphe G contient une clique de taille k, alors le nombre chromatique de G est au moins égal à k. Comme le pentagone (cycle de longueur cinq) le montre, le nombre chromatique peut dépasser la taille de la clique maximum. En effet, il

existe des graphes ayant des nombres chromatiques arbitrairement grands qui ne contiennent pas de graphe K_3. Cela motive la définition suivante, qui est due à Berge [1961,1962] :

Définition 16.19. *Un graphe G est* **parfait** *si $\chi(H) = \omega(H)$ pour tout sous-graphe induit H de G, où $\chi(H)$ est le nombre chromatique et $\omega(H)$ est la cardinalité maximum d'une clique dans H.*

Une conséquence immédiate est que le problème de décider si un graphe parfait donné est de nombre chromatique égal à k a une bonne caractérisation (il appartient à *NP* ∩ *coNP*). Quelques exemples de graphes parfaits sont donnés à l'exercice 15. Un algorithme polynomial pour reconnaître les graphes parfaits a été trouvé par Chudnovsky *et al.* [2005].

Berge [1961] a conjecturé qu'un graphe est parfait si et seulement s'il ne contient ni un cycle impair de longueur au moins cinq ni le complémentaire d'un tel cycle comme sous-graphe induit. Cette proposition appelée aussi théorème fort des graphes parfaits a été prouvée par Chudnovsky *et al.* [2006]. Trente ans plus tôt, Lovász [1972] avait prouvé la proposition plus faible qu'un graphe est parfait si et seulement si son complémentaire est parfait. Cela est appelé le théorème faible des graphes parfaits. Pour le prouver, nous avons besoin du lemme suivant :

Lemme 16.20. *Soient G un graphe parfait et $x \in V(G)$. Alors le graphe $G' := (V(G) \dot\cup \{y\}, E(G) \dot\cup \{(y,v) : v \in \{x\} \cup \Gamma(x)\})$, obtenu de G en ajoutant un nouveau sommet y qui est relié à x et à tous les voisins de x, est parfait.*

Preuve. Par induction sur $|V(G)|$. Le cas $|V(G)| = 1$ est trivial puisque K_2 est parfait. Considérons maintenant un graphe parfait G ayant au moins deux sommets. Soit $x \in V(G)$ et soit G' le graphe obtenu en ajoutant un nouveau sommet y adjacent à x et à tous ses voisins. Il suffit de prouver que $\omega(G') = \chi(G')$ puisque, pour un sous-graphe propre H de G', c'est une conséquence de l'hypothèse d'induction : soit H est un sous-graphe de G et est donc parfait, soit il provient d'un sous-graphe propre de G en ajoutant un sommet y comme ci-dessus.

Comme on peut aisément colorer G' à l'aide de $\chi(G) + 1$ couleurs, on peut supposer que $\omega(G') = \omega(G)$. Alors x n'est pas contenu dans une clique maximum de G. Soit f une coloration des sommets de G à l'aide de $\chi(G)$ couleurs, et soit $X := \{v \in V(G) : f(v) = f(x)\}$. On a $\omega(G - X) = \chi(G - X) = \chi(G) - 1 = \omega(G) - 1$ et ainsi $\omega(G - (X \setminus \{x\})) = \omega(G) - 1$ (car x n'appartient pas à une clique maximum de G). Comme $(X \setminus \{x\}) \cup \{y\} = V(G') \setminus V(G - (X \setminus \{x\}))$ est un ensemble stable, on a

$$\chi(G') = \chi(G - (X \setminus \{x\})) + 1 = \omega(G - (X \setminus \{x\})) + 1 = \omega(G) = \omega(G').$$

□

Théorème 16.21. (Lovász [1972], Fulkerson [1972], Chvátal [1975]) *Pour un graphe simple G, les affirmations suivantes sont équivalentes :*

(a) *G est parfait.*
(b) *Le complémentaire de G est parfait.*
(c) *Le polytope des ensembles stables, c.-à-d. l'enveloppe convexe des vecteurs d'incidence des ensembles stables de G, est décrit par :*

$$\left\{ x \in \mathbb{R}_+^{V(G)} : \sum_{v \in S} x_v \leq 1 \text{ pour toutes les cliques } S \text{ de } G \right\}. \tag{16.3}$$

Preuve. Nous prouvons (a)⇒(c)⇒(b). Cela est suffisant, car, en appliquant (a)⇒(b) au complémentaire de G, on obtient (b)⇒(a).

(a)⇒(c) : évidemment le polytope des ensembles stables est contenu dans (16.3). Pour prouver l'autre inclusion, considérons un vecteur rationnel x dans le polytope (16.3) ; on peut écrire $x_v = \frac{p_v}{q}$, où $q \in \mathbb{N}$ et $p_v \in \mathbb{Z}_+$ pour $v \in V(G)$. Remplaçons chaque sommet v par une clique de taille p_v ; c.-à-d. considérons G' défini par

$$\begin{aligned} V(G') &:= \{(v,i) : v \in V(G),\ 1 \leq i \leq p_v\}, \\ E(G') &:= \{((v,i),(v,j)) : v \in V(G),\ 1 \leq i < j \leq p_v\} \cup \\ &\quad \{((v,i),(w,j)) : (v,w) \in E(G),\ 1 \leq i \leq p_v,\ 1 \leq j \leq p_w\}. \end{aligned}$$

Le lemme 16.20 implique que G' est parfait. Pour une clique arbitraire X' de G' notons $X := \{v \in V(G) : (v,i) \in X' \text{ pour un certain } i\}$ sa projection sur G (qui est encore une clique) ; on a

$$|X'| \ \leq \ \sum_{v \in X} p_v \ = \ q \sum_{v \in X} x_v \ \leq \ q.$$

Ainsi $\omega(G') \leq q$. Puisque G' est parfait, il a donc une coloration des sommets f à l'aide d'au plus q couleurs. Pour $v \in V(G)$ et $i = 1, \ldots, q$ posons $a_{i,v} := 1$ si $f((v,j)) = i$ pour un certain j et $a_{i,v} := 0$ sinon. Alors $\sum_{i=1}^{q} a_{i,v} = p_v$ pour tout $v \in V(G)$ et ainsi

$$x \ = \ \left(\frac{p_v}{q} \right)_{v \in V(G)} \ = \ \frac{1}{q} \sum_{i=1}^{q} a_i$$

est une combinaison convexe des vecteurs d'incidence des ensembles stables, où $a_i = (a_{i,v})_{v \in V(G)}$.

(c)⇒(b) : nous montrons par induction sur $|V(G)|$ que si (16.3) est entier alors le complémentaire de G est parfait. Puisque les graphes ayant moins de trois sommets sont parfaits, considérons un graphe G avec $|V(G)| \geq 3$ tel que (16.3) soit entier.

Nous devons montrer que l'ensemble des sommets de tout sous-graphe induit H de G peut être partitionné en $\alpha(H)$ cliques, où $\alpha(H)$ est la taille d'un ensemble stable maximum de H. Pour des sous-graphes propres H, c'est une conséquence de l'hypothèse d'induction, puisque (d'après le théorème 5.13) toute face du polytope entier (16.3) est entière, en particulier la face définie par les hyperplans supports $x_v = 0$ ($v \in V(G) \setminus V(H)$).

Il reste donc à prouver que $V(G)$ peut être partitionné en $\alpha(G)$ cliques. L'équation $\mathbb{1}x = \alpha(G)$ définit un hyperplan support de (16.3), donc

$$\left\{ x \in \mathbb{R}_+^{V(G)} : \sum_{v \in S} x_v \leq 1 \text{ pour toute clique } S \text{ de } G, \sum_{v \in V(G)} x_v = \alpha(G) \right\} \tag{16.4}$$

est une face de (16.3). Cette face est contenue dans des facettes, qui ne peuvent pas être toutes de la forme $\{x \in (16.3) : x_v = 0\}$ pour un certain v (sinon l'origine appartiendrait à l'intersection). Il existe donc une clique S de G telle que $\sum_{v \in S} x_v = 1$ pour tout x dans (16.4). Ainsi cette clique S intersecte chaque ensemble stable maximum de G. Par l'hypothèse d'induction, l'ensemble des sommets de $G - S$ peut être maintenant partitionné en $\alpha(G - S) = \alpha(G) - 1$ cliques. L'ajout de S termine la preuve. □

Cette preuve est due à Lovász [1979b]. Le système d'inégalités qui définit (16.3) est TDI pour les graphes parfaits (exercice 16). Avec un peu plus de travail on peut prouver que, pour les graphes parfaits, le PROBLÈME DE LA COLORATION DES SOMMETS, le PROBLÈME DE L'ENSEMBLE STABLE DE POIDS MAXIMUM et le PROBLÈME DE LA CLIQUE DE POIDS MAXIMUM peuvent être résolus en temps fortement polynomial. Bien que ces problèmes soient tous *NP*-difficiles pour les graphes généraux (théorème 15.23, corollaire 15.24, théorème 16.14(b)), il existe un nombre (appelé fonction-thêta du graphe complémentaire, introduit par Lovász [1979a]) qui est toujours compris entre la taille d'une clique maximum et le nombre chromatique, et qui peut être calculé en temps polynomial pour les graphes généraux en utilisant la MÉTHODE DES ELLIPSOÏDES. Les détails sont un peu compliqués ; voir Grötschel, Lovász et Schrijver [1988].

Un des problèmes les plus connus de la théorie des graphes est le problème des quatre couleurs : est-il vrai que toute carte planaire peut être colorée avec quatre couleurs de telle façon que deux pays qui ont une frontière commune n'aient pas la même couleur ? Si on considère les pays comme des régions et que l'on passe au graphe dual planaire, cela équivaut à demander si tout graphe planaire possède une coloration des sommets à l'aide de quatre couleurs. Appel et Haken [1977] et Appel, Haken et Koch [1977] ont prouvé que cela est effectivement vrai : tout graphe planaire a son nombre chromatique au plus égal à 4. Pour une preuve plus simple du théorème des quatre couleurs (qui est cependant fondée sur une vérification de cas à l'aide d'un ordinateur), voir Robertson *et al.* [1997]. Nous prouvons le résultat plus faible suivant, connu sous le nom de théorème des cinq couleurs :

Théorème 16.22. (Heawood [1890]) *Tout graphe planaire a une coloration des sommets à l'aide d'au plus cinq couleurs, et une telle coloration peut être trouvée en temps polynomial.*

Preuve. Par induction sur $|V(G)|$. On peut supposer que G est simple, et on fixe une représentation plane arbitraire $\Phi = \big(\psi, (J_e)_{e \in E(G)}\big)$ de G. D'après le corollaire 2.33, G possède un sommet v de degré cinq ou moins. Par l'hypothèse d'induction, $G - v$ a une coloration des sommets f à l'aide d'au plus cinq couleurs. On peut supposer que v est de degré 5 et que tous ses voisins ont des couleurs différentes ; sinon on peut facilement étendre la coloration à G.

Soient w_1, w_2, w_3, w_4, w_5 les voisins de v dans l'ordre cyclique dans lequel les courbes polygonales $J_{(v,w_i)}$ quittent v.

Nous affirmons d'abord qu'il n'existe pas de chaînes sommet-disjointes P de w_1 à w_3 et Q de w_2 à w_4 dans $G - v$. Afin de prouver cela, soit P une chaîne de w_1 à w_3, et soit C le cycle de G constitué de P et des arêtes $(v, w_1), (v, w_3)$. D'après le théorème 2.30 $\mathbb{R}^2 \setminus \bigcup_{e \in E(C)} J_e$ se divise en deux régions connexes, et v est sur la frontière de ces deux régions. Ainsi w_2 et w_4 appartiennent à des régions différentes de cet ensemble, ce qui implique que toute chaîne de w_2 à w_4 de $G - v$ doit contenir un sommet de C.

Soit X la composante connexe du graphe $G[\{x \in V(G) \setminus \{v\} : f(x) \in \{f(w_1), f(w_3)\}\}]$ qui contient w_1. Si X ne contient pas w_3, on peut échanger les couleurs dans X et après étendre la coloration à G en colorant v avec l'ancienne couleur de w_1. On peut donc supposer qu'il existe une chaîne P de w_1 à w_3 contenant seulement des sommets colorés avec $f(w_1)$ ou $f(w_3)$.

De façon analogue, nous avons terminé s'il n'existe pas de chaîne de w_2 à w_4 Q contenant seulement des sommets colorés avec $f(w_2)$ ou $f(w_4)$. Mais l'hypothèse contraire signifie qu'il existe des chaînes sommet-disjointes P de w_1 à w_3 et Q de w_2 à w_4 dans $G - v$, une contradiction. □

Ainsi c'est un second problème *NP*-difficile qui a un algorithme d'approximation absolue. En effet, le théorème des quatre couleurs implique que le nombre chromatique d'un graphe planaire non biparti peut être seulement égal à 3 ou 4. En utilisant l'algorithme polynomial de Robertson *et al.* [1996], qui permet de colorer tout graphe planaire à l'aide de quatre couleurs, on obtient un algorithme d'approximation absolue qui utilise au plus une couleur de plus que nécessaire.

Fürer et Raghavachari [1994] ont découvert un troisième problème naturel qui peut être approximé à une erreur additive près égale à un : étant donné un graphe non orienté, ils recherchent un arbre couvrant dont le degré maximum soit minimum parmi tous les arbres couvrants (le problème est une généralisation du PROBLÈME DE LA CHAÎNE HAMILTONIENNE et il est donc *NP*-difficile). Leur algorithme s'étend également au cas général correspondant au PROBLÈME DE L'ARBRE DE STEINER : étant donné un ensemble $T \subseteq V(G)$, trouver un arbre S dans G avec $V(T) \subseteq V(S)$ tel que le degré maximum de S soit minimum. Singh et Lau [2007] ont trouvé une extension aux arbres couvrants de poids minimum et de degrés bornés.

D'autre part, le théorème suivant précise que pour de nombreux problèmes il n'existe pas d'algorithmes d'approximation absolue à moins que $P = NP$:

Proposition 16.23. *Soient $\mathcal{F}$ et $\mathcal{F}'$ des familles (infinies) d'ensembles finis, et soit $\mathcal{P}$ le problème d'optimisation suivant : étant donné un ensemble $E \in \mathcal{F}$ ainsi qu'une une fonction $c : E \to \mathbb{Z}$, trouver un ensemble $F \subseteq E$ tel que $F \in \mathcal{F}'$ et que $c(F)$ soit minimum (ou conclure qu'un tel ensemble F n'existe pas).*

Alors $\mathcal{P}$ a un algorithme d'approximation absolue si et seulement si $\mathcal{P}$ peut être résolu en temps polynomial.

Preuve. Supposons qu'il existe un algorithme polynomial A et un entier k tels que

$$|A((E,c)) - \mathrm{OPT}((E,c))| \;\leq\; k$$

pour toutes les instances (E,c) de $\mathcal{P}$. Nous montrons comment résoudre $\mathcal{P}$ de manière exacte en temps polynomial.

Étant donné une instance (E,c) de $\mathcal{P}$, on construit une nouvelle instance (E,c'), telle que $c'(e) := (k+1)c(e)$ pour tout $e \in E$. Évidemment les solutions optimales restent les mêmes. Mais si on applique maintenant A à la nouvelle instance, on a

$$|A((E,c')) - \mathrm{OPT}((E,c'))| \;\leq\; k$$

et ainsi $A((E,c')) = \mathrm{OPT}((E,c'))$. □

Des exemples de tels problèmes sont le PROBLÈME DE MINIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE, le PROBLÈME DE MAXIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE (multiplier c par -1), ainsi que tous les problèmes de la liste du paragraphe 13.1.

16.4 Schémas d'approximation

Reportons-nous à l'algorithme d'approximation absolue pour le PROBLÈME DE LA COLORATION DES ARÊTES présenté au paragraphe précédent. Il implique également une garantie de performance relative : puisque l'on peut aisément décider si le nombre arête-chromatique est égal à 1 ou 2 (proposition 16.15), le théorème de Vizing fournit un algorithme d'approximation de facteur $\frac{4}{3}$. D'autre part, le théorème 16.14(a) implique qu'aucun algorithme d'approximation de facteur k ne peut exister pour $k < \frac{4}{3}$ (à moins que $P = NP$).

Ainsi l'existence d'un algorithme d'approximation absolue n'implique pas l'existence d'un algorithme d'approximation de facteur k pour tout $k > 1$. Nous rencontrerons une situation similaire avec le PROBLÈME DU BIN-PACKING au chapitre 18. Cette observation suggère la définition suivante :

Définition 16.24. *Soit $\mathcal{P}$ un problème d'optimisation avec des poids non négatifs. Un* **algorithme d'approximation asymptotique de facteur k** *pour $\mathcal{P}$ est un algorithme polynomial A pour $\mathcal{P}$ pour lequel il existe une constante c telle que*

$$\frac{1}{k}\,\mathrm{OPT}(I) - c \;\leq\; A(I) \;\leq\; k\,\mathrm{OPT}(I) + c$$

pour toutes les instances I de $\mathcal{P}$. On dit également que A a un **rapport de performance asymptotique** *égal à k.*

Le **rapport d'approximation (asymptotique)** d'un problème d'optimisation $\mathcal{P}$ avec des poids non négatifs est défini comme étant l'infimum de tous les nombres k pour lesquels il existe un algorithme d'approximation (asymptotique) de facteur k pour $\mathcal{P}$, ou égal à ∞ s'il n'existe aucun algorithme d'approximation (asymptotique).

Par exemple, le PROBLÈME DE LA COLORATION DES ARÊTES mentionné ci-dessus a un rapport d'approximation égal à $\frac{4}{3}$ (à moins que $P = NP$), mais un rapport

d'approximation asymptotique égal à 1 (pas seulement pour les graphes simples ; voir Sanders et Steurer [2005]). Les problèmes d'optimisation de rapport d'approximation (asymptotique) égal à 1 ont un intérêt particulier. Pour ces problèmes nous introduisons la notion suivante :

Définition 16.25. *Soit $\mathcal{P}$ un problème d'optimisation avec des poids non négatifs. Un* **schéma d'approximation** *pour $\mathcal{P}$ est un algorithme A acceptant comme input une instance I de $\mathcal{P}$ et un $\epsilon > 0$ tels que, pour tout ϵ fixé, A est un algorithme d'approximation de facteur $(1+\epsilon)$ pour $\mathcal{P}$.*

Un **schéma d'approximation asymptotique** *pour $\mathcal{P}$ est une paire d'algorithmes (A, A') ayant les propriétés suivantes : A' est un algorithme polynomial acceptant un nombre $\epsilon > 0$ comme input et calculant un nombre c_ϵ. A accepte une instance I de $\mathcal{P}$ et un $\epsilon > 0$ comme input, et son output est constitué d'une solution réalisable pour I vérifiant*

$$\frac{1}{1+\epsilon}\,\mathrm{OPT}(I) - c_\epsilon \;\leq\; A(I,\epsilon) \;\leq\; (1+\epsilon)\,\mathrm{OPT}(I) + c_\epsilon.$$

Pour chaque ϵ fixé, le temps de calcul de A est polynomialement borné par rapport à $\mathrm{taille}(I)$.

Un schéma d'approximation (asymptotique) est appelé un **schéma d'approximation (asymptotique) entièrement polynomial** *si le temps de calcul ainsi que la taille maximum de tout nombre apparaissant dans le calcul sont bornés par un polynôme par rapport à* $\mathrm{taille}(I) + \mathrm{taille}(\epsilon) + \frac{1}{\epsilon}$.

Dans certains ouvrages, on trouve les abréviations PTAS pour schéma d'approximation (polynomial) et FPAS pour schéma d'approximation entièrement polynomial.

Si on met à part les algorithmes d'approximation absolue, un schéma d'approximation entièrement polynomial peut être considéré comme le meilleur algorithme que l'on puisse espérer lorsque l'on est face à un problème d'optimisation *NP*-difficile, en tout cas si le coût d'une solution réalisable est un entier non négatif (ce que l'on peut supposer dans de nombreux cas sans perte de généralité) :

Proposition 16.26. *Soit $\mathcal{P} = (X, (S_x)_{x\in X}, c, \mathrm{but})$ un problème d'optimisation où les valeurs de c sont des entiers non négatifs. Soit A un algorithme qui, étant donné une instance I de $\mathcal{P}$ et un nombre $\epsilon > 0$, calcule une solution réalisable de I telle que*

$$\frac{1}{1+\epsilon}\,\mathrm{OPT}(I) \;\leq\; A(I,\epsilon) \;\leq\; (1+\epsilon)\,\mathrm{OPT}(I)$$

et dont le temps de calcul soit borné par un polynôme par rapport à $\mathrm{taille}(I) + \mathrm{taille}(\epsilon)$. *Alors $\mathcal{P}$ peut être résolu de façon exacte en temps polynomial.*

Preuve. Étant donné une instance I, on exécute d'abord A sur $(I, 1)$. On pose $\epsilon := \frac{1}{1+2A(I,1)}$ et on observe que $\epsilon\,\mathrm{OPT}(I) < 1$. On exécute alors A sur (I, ϵ). Puisque $\mathrm{taille}(\epsilon)$ est borné polynomialement par rapport à $\mathrm{taille}(I)$, cette procédure constitue un algorithme polynomial. Si $\mathcal{P}$ est un problème de minimisation, on a

$$A(I,\epsilon) \;\leq\; (1+\epsilon)\,\mathrm{OPT}(I) \;<\; \mathrm{OPT}(I)+1,$$

ce qui, puisque c est entier, implique l'optimalité. De manière similaire, si $\mathcal{P}$ est un problème de maximisation, on a

$$A(I,\epsilon) \;\geq\; \frac{1}{1+\epsilon}\,\mathrm{OPT}(I) \;>\; (1-\epsilon)\,\mathrm{OPT}(I) \;>\; \mathrm{OPT}(I)-1.$$

□

Malheureusement, un schéma d'approximation entièrement polynomial n'existe que pour un nombre restreint de problèmes (voir théorème 17.11). De plus, notons que même l'existence d'un schéma d'approximation entièrement polynomial n'implique pas l'existence d'un algorithme d'approximation absolue ; le PROBLÈME DU SAC À DOS en est un exemple.

Aux chapitres 17 et 18, nous présenterons deux problèmes (SAC À DOS et BIN-PACKING) qui ont, respectivement, un schéma d'approximation entièrement polynomial et un schéma d'approximation asymptotique entièrement polynomial. Pour de nombreux problèmes, les deux types de schémas d'approximation coïncident :

Théorème 16.27. (Papadimitriou et Yannakakis [1993]) *Soit $\mathcal{P}$ un problème d'optimisation avec des poids non négatifs. Supposons que, pour toute constante k, il existe un algorithme polynomial qui permet de décider si une instance donnée a une valeur optimale au plus égale à k, et, si c'est le cas, trouver une solution optimale.*

Alors $\mathcal{P}$ a un schéma d'approximation si et seulement si $\mathcal{P}$ a un schéma d'approximation asymptotique.

Preuve. La condition suffisante est évidente. Supposons donc que $\mathcal{P}$ ait un schéma d'approximation asymptotique (A, A'). Nous décrivons un schéma d'approximation pour $\mathcal{P}$.

Soit un $\epsilon > 0$ fixé ; on peut supposer $\epsilon < 1$. On pose $\epsilon' := \frac{\epsilon-\epsilon^2}{2+\epsilon+\epsilon^2} < \frac{\epsilon}{2}$ et on exécute d'abord A' sur l'input ϵ', ce qui fournit une constante $c_{\epsilon'}$.

Étant donné une instance I, on teste ensuite si $\mathrm{OPT}(I)$ est au plus égal à $\frac{2c_{\epsilon'}}{\epsilon}$. Il s'agit d'une constante pour chaque ϵ fixé, on peut donc décider cela en temps polynomial et trouver une solution optimale si $\mathrm{OPT}(I) \leq \frac{2c_{\epsilon'}}{\epsilon}$.

Sinon on applique A à I et ϵ' et on obtient une solution de valeur V, telle que

$$\frac{1}{1+\epsilon'}\;\mathrm{OPT}(I) - c_{\epsilon'} \;\leq\; V \;\leq\; (1+\epsilon')\;\mathrm{OPT}(I) + c_{\epsilon'}.$$

Nous affirmons que cette solution est suffisamment bonne. En effet, on a $c_{\epsilon'} < \frac{\epsilon}{2}\,\mathrm{OPT}(I)$, ce qui implique

$$V \;\leq\; (1+\epsilon')\;\mathrm{OPT}(I) + c_{\epsilon'} \;<\; \left(1+\frac{\epsilon}{2}\right)\mathrm{OPT}(I) + \frac{\epsilon}{2}\,\mathrm{OPT}(I) \;=\; (1+\epsilon)\,\mathrm{OPT}(I)$$

et

$$\begin{aligned} V &\geq \frac{1}{(1+\epsilon')}\,\mathrm{OPT}(I) - \frac{\epsilon}{2}\,\mathrm{OPT}(I) \\ &= \frac{2+\epsilon+\epsilon^2}{2+2\epsilon}\,\mathrm{OPT}(I) - \frac{\epsilon}{2}\,\mathrm{OPT}(I) \\ &= \left(\frac{1}{1+\epsilon} + \frac{\epsilon}{2}\right)\mathrm{OPT}(I) - \frac{\epsilon}{2}\,\mathrm{OPT}(I) \\ &= \frac{1}{1+\epsilon}\,\mathrm{OPT}(I). \end{aligned}$$

□

Ainsi la définition d'un schéma d'approximation asymptotique n'a de sens que pour les problèmes (tels que ceux du bin-packing ou de coloration) dont la restriction à une valeur optimale constante est encore difficile. Pour de nombreux problèmes, cette restriction peut se résoudre en temps polynomial à l'aide d'une sorte d'énumération complète.

16.5 Satisfaisabilité maximum

Le PROBLÈME DE LA SATISFAISABILITÉ était notre premier problème *NP*-complet. Analysons maintenant le problème d'optimisation correspondant :

SATISFAISABILITÉ MAXIMUM (MAX-SAT)	
Instance	Un ensemble X de variables, une famille $\mathcal{Z}$ de clauses sur X, et une fonction poids $c : \mathcal{Z} \to \mathbb{R}_+$.
Tâche	Trouver un assignement T de X tel que le poids total des clauses de $\mathcal{Z}$, qui sont satisfaites par T, soit maximum.

Comme nous le verrons, l'approximation de MAX-SAT est un bon exemple (et historiquement l'un des premiers) de l'utilisation de la méthode probabiliste.

Considérons d'abord l'algorithme randomisé suivant : poser indépendamment chaque variable à *vrai* avec probabilité $\frac{1}{2}$. Évidemment, cet algorithme satisfait chaque clause Z avec probabilité $1 - 2^{-|Z|}$.

Notons r la variable aléatoire qui est *vrai* avec probabilité $\frac{1}{2}$ et *faux* sinon, et soit $R = (r, r, \ldots, r)$ la variable aléatoire uniformément distribuée sur tous les assignements. Si on note $c(T)$ le poids total des clauses satisfaites par l'assignement T, l'espérance du poids total des clauses satisfaites par R est

$$\begin{aligned} \mathrm{Exp}\,(c(R)) &= \sum_{Z\in\mathcal{Z}} c(Z)\,\mathrm{Prob}(R \text{ satisfait } Z) \\ &= \sum_{Z\in\mathcal{Z}} c(Z)\left(1 - 2^{-|Z|}\right) \qquad (16.5) \\ &\geq \left(1 - 2^{-p}\right)\sum_{Z\in\mathcal{Z}} c(Z), \end{aligned}$$

où $p := \min_{Z \in \mathcal{Z}} |Z|$. Exp et Prob désignent l'espérance et la probabilité.

Puisque l'optimum ne peut excéder $\sum_{Z \in \mathcal{Z}} c(Z)$, R fournit en espérance une solution à un facteur $\frac{1}{1-2^{-p}}$ près de l'optimum. Mais ce que nous aimerions réellement avoir, c'est un algorithme d'approximation déterministe. En fait, on peut modifier notre algorithme randomisé (trivial) en un algorithme déterministe tout en préservant la garantie de performance. Cette étape est souvent appelée dérandomisation.

Établissons l'assignement étape par étape. Supposons que $X = \{x_1, \ldots, x_n\}$ et que nous avons déjà fixé un assignement T pour $x_1, \ldots, x_k$ ($0 \leq k < n$). Si nous fixons maintenant $x_{k+1}, \ldots, x_n$ aléatoirement, en posant chaque variable indépendamment à *vrai* avec probabilité $\frac{1}{2}$, nous satisferons les clauses avec une espérance de poids total égale à $e_0 = c(T(x_1), \ldots, T(x_k), r, \ldots, r)$. Si on fixe x_{k+1} à *vrai* (*faux*), et ensuite $x_{k+2}, \ldots, x_n$ aléatoirement, les clauses satisfaites auront une espérance de poids total égale à e_1 (e_2, respectivement). e_1 et e_2 peuvent être considérées comme des espérances conditionnelles. Évidemment $e_0 = \frac{e_1+e_2}{2}$, donc au moins l'une des espérances e_1, e_2 doit être supérieure ou égale à e_0. On pose x_{k+1} à *vrai* si $e_1 \geq e_2$ et à *faux* sinon. Cela est parfois appelé la méthode des probabilités conditionnelles.

ALGORITHME DE JOHNSON POUR MAX-SAT

Input Un ensemble $X = \{x_1, \ldots, x_n\}$ de variables, une famille $\mathcal{Z}$ de clauses sur X, et une fonction poids $c : \mathcal{Z} \to \mathbb{R}_+$.

Output Un assignement $T : X \to \{\textit{vrai}, \textit{faux}\}$.

① **For** $k := 1$ **to** n **do** :
If $\mathrm{Exp}(c(T(x_1), \ldots, T(x_{k-1}), \textit{vrai}, r, \ldots, r))$
$\geq \mathrm{Exp}(c(T(x_1), \ldots, T(x_{k-1}), \textit{faux}, r, \ldots, r))$
then poser $T(x_k) := \textit{vrai}$
else poser $T(x_k) := \textit{faux}$.

Les espérances peuvent être aisément calculées à l'aide de (16.5).

Théorème 16.28. (Johnson [1974]) *L'*ALGORITHME DE JOHNSON POUR MAX-SAT *est un algorithme d'approximation de facteur* $\frac{1}{1-2^{-p}}$ *pour* MAX-SAT*, où* p *est la cardinalité minimum d'une clause.*

Preuve. Définissons l'espérance conditionnelle

$$s_k := \mathrm{Exp}(c(T(x_1), \ldots, T(x_k), r, \ldots, r))$$

pour $k = 0, \ldots, n$. Observons que $s_n = c(T)$ est le poids total des clauses satisfaites par notre algorithme, tandis que $s_0 = \mathrm{Exp}(c(R)) \geq (1 - 2^{-p}) \sum_{Z \in \mathcal{Z}} c(Z)$ d'après (16.5).

De plus, $s_i \geq s_{i-1}$ d'après le choix de $T(x_i)$ à l'étape ① (pour $i = 1, \ldots, n$). Donc $s_n \geq s_0 \geq (1 - 2^{-p}) \sum_{Z \in \mathcal{Z}} c(Z)$. Puisque l'optimum est au plus égal à $\sum_{Z \in \mathcal{Z}} c(Z)$, la preuve est terminée. □

Puisque $p \geq 1$, on a un algorithme d'approximation de facteur 2. Cependant, cela n'est pas très intéressant, car il existe un algorithme d'approximation de facteur 2 beaucoup plus simple : fixer les variables soit toutes à *vrai*, soit toutes à *faux*, suivant ce qui est le mieux. Cependant, Chen, Friesen et Zheng [1999] ont montré que l'ALGORITHME DE JOHNSON POUR MAX-SAT est en fait un algorithme d'approximation de facteur $\frac{3}{2}$.

S'il n'y a pas de clauses constituées d'un seul élément ($p \geq 2$), c'est un algorithme d'approximation de facteur $\frac{4}{3}$ (d'après le théorème 16.28). Si $p \geq 3$, c'est un algorithme d'approximation de facteur $\frac{8}{7}$.

Yannakakis [1994] a trouvé un algorithme d'approximation de facteur $\frac{4}{3}$, pour le cas général, qui utilise des techniques de flots dans les réseaux. Nous décrirons un algorithme d'approximation de facteur $\frac{4}{3}$ plus récent et plus simple dû à Goemans et Williamson [1994].

Il est facile de traduire MAX-SAT en un PL en nombres entiers : Si on note l'ensemble des variables $X = \{x_1, \ldots, x_n\}$, l'ensemble des clauses $\mathcal{Z} = \{Z_1, \ldots, Z_m\}$, et les poids $c_1, \ldots, c_m$, MAX-SAT correspond au PL suivant :

$$\begin{array}{lll} \max & \displaystyle\sum_{j=1}^{m} c_j z_j & \\ \text{s.c.} & z_j \leq \displaystyle\sum_{i:x_i \in Z_j} y_i + \sum_{i:\overline{x_i} \in Z_j} (1 - y_i) & (j = 1, \ldots, m) \\ & y_i, z_j \in \{0,1\} & (i = 1, \ldots, n,\ j = 1, \ldots, m). \end{array}$$

Ici $y_i = 1$ signifie que la variable x_i est fixée à *vrai*, et $z_j = 1$ signifie que la clause Z_j est satisfaite. Considérons maintenant la relaxation linéaire :

$$\begin{array}{lll} \max & \displaystyle\sum_{j=1}^{m} c_j z_j & \\ \text{s.c.} & z_j \leq \displaystyle\sum_{i:x_i \in Z_j} y_i + \sum_{i:\overline{x_i} \in Z_j} (1 - y_i) & (j = 1, \ldots, m) \\ & y_i \leq 1 & (i = 1, \ldots, n) \\ & y_i \geq 0 & (i = 1, \ldots, n) \\ & z_j \leq 1 & (j = 1, \ldots, m) \\ & z_j \geq 0 & (j = 1, \ldots, m). \end{array} \tag{16.6}$$

Soit (y^*, z^*) une solution optimale de (16.6). Fixons maintenant indépendamment chaque variable x_i à *vrai* avec probabilité y_i^*. Cette étape est connue sous le nom d'arrondi aléatoire, une technique qui a été introduite par Raghavan et Thompson [1987]. La méthode précédente constitue un autre algorithme randomisé pour MAX-SAT, qui peut être dérandomisé comme ci-dessus. Soit r_p la variable aléatoire qui est *vrai* avec probabilité p et *faux* sinon.

ALGORITHME DE GOEMANS-WILLIAMSON POUR MAX-SAT

Input Un ensemble $X = \{x_1, \dots, x_n\}$ de variables, une famille $\mathcal{Z}$ de clauses sur X, et une fonction poids $c : \mathcal{Z} \to \mathbb{R}_+$.

Output Un assignement $T : X \to \{\textit{vrai}, \textit{faux}\}$.

① Résoudre le PL (16.6) ; soit (y^*, z^*) une solution optimale.

② **For** $k := 1$ **to** n **do** :
If $\mathrm{Exp}(c(T(x_1), \dots, T(x_{k-1}), \textit{vrai}, r_{y^*_{k+1}}, \dots, r_{y^*_n})$
$\geq \mathrm{Exp}(c(T(x_1), \dots, T(x_{k-1}), \textit{faux}, r_{y^*_{k+1}}, \dots, r_{y^*_n})$
then poser $T(x_k) := \textit{vrai}$
else poser $T(x_k) := \textit{faux}$.

Théorème 16.29. (Goemans et Williamson [1994]) *L'*ALGORITHME DE GOEMANS-WILLIAMSON POUR MAX-SAT *est un algorithme d'approximation de facteur* $\frac{1}{1-\left(1-\frac{1}{q}\right)^q}$*, où* q *est la cardinalité maximum d'une clause.*

Preuve. Écrivons

$$s_k \;:=\; \mathrm{Exp}(c(T(x_1), \dots, T(x_k), r_{y^*_{k+1}}, \dots, r_{y^*_n}))$$

pour $k = 0, \dots, n$. On a encore $s_i \geq s_{i-1}$ pour $i = 1, \dots, n$ et $s_n = c(T)$ est le poids total des clauses satisfaites par notre algorithme. Il reste donc à estimer $s_0 = \mathrm{Exp}(c(R_{y^*}))$, où $R_{y^*} = (r_{y^*_1}, \dots, r_{y^*_n})$.

Pour $j = 1, \dots, m$, la probabilité que la clause Z_j soit satisfaite par R_{y^*} est égale à

$$1 - \left(\prod_{i:x_i \in Z_j} (1 - y^*_i)\right) \cdot \left(\prod_{i:\overline{x_i} \in Z_j} y^*_i\right).$$

Puisque la moyenne géométrique est toujours inférieure ou égale à la moyenne arithmétique, cette probabilité est au moins égale à

$$\begin{aligned}
& 1 - \left(\frac{1}{|Z_j|}\left(\sum_{i:x_i \in Z_j} (1 - y^*_i) + \sum_{i:\overline{x_i} \in Z_j} y^*_i\right)\right)^{|Z_j|} \\
= \;& 1 - \left(1 - \frac{1}{|Z_j|}\left(\sum_{i:x_i \in Z_j} y^*_i + \sum_{i:\overline{x_i} \in Z_j} (1 - y^*_i)\right)\right)^{|Z_j|} \\
\geq \;& 1 - \left(1 - \frac{z^*_j}{|Z_j|}\right)^{|Z_j|} \\
\geq \;& \left(1 - \left(1 - \frac{1}{|Z_j|}\right)^{|Z_j|}\right) z^*_j.
\end{aligned}$$

Pour prouver la dernière inégalité, observons que pour tout $0 \leq a \leq 1$ et pour tout $k \in \mathbb{N}$, l'inégalité

$$1 - \left(1 - \frac{a}{k}\right)^k \geq a\left(1 - \left(1 - \frac{1}{k}\right)^k\right)$$

est vérifiée : les deux côtés de l'inégalité sont égaux pour $a \in \{0, 1\}$, et le côté gauche (en tant que fonction de a) est concave, alors que le côté droit est linéaire.

On a donc

$$\begin{aligned} s_0 = \mathrm{Exp}(c(R_{y^*})) &= \sum_{j=1}^{m} c_j \,\mathrm{Prob}(R_{y^*} \text{ satisfait } Z_j) \\ &\geq \sum_{j=1}^{m} c_j \left(1 - \left(1 - \frac{1}{|Z_j|}\right)^{|Z_j|}\right) z_j^* \\ &\geq \left(1 - \left(1 - \frac{1}{q}\right)^q\right) \sum_{j=1}^{m} c_j z_j^* \end{aligned}$$

(observons que la suite $\left(\left(1 - \frac{1}{k}\right)^k\right)_{k\in\mathbb{N}}$ est monotone croissante et converge vers $\frac{1}{e}$). Puisque l'optimum est inférieur ou égal à $\sum_{j=1}^{m} z_j^* c_j$, la valeur optimale de la relaxation linéaire, la preuve est terminée. □

Puisque $\left(1 - \frac{1}{q}\right)^q < \frac{1}{e}$, on a un algorithme d'approximation de facteur $\frac{e}{e-1}$ ($\frac{e}{e-1}$ vaut environ $1,582$).

Nous avons maintenant deux algorithmes similaires qui se comportent différemment : le premier est meilleur pour les clauses longues, alors que le second est meilleur pour les clauses courtes. Il est ainsi naturel de les combiner :

Théorème 16.30. (Goemans et Williamson [1994]) *L'algorithme suivant est un algorithme d'approximation de facteur $\frac{4}{3}$ pour* MAX-SAT *: exécuter l'*ALGORITHME DE JOHNSON POUR MAX-SAT *et l'*ALGORITHME DE GOEMANS-WILLIAMSON POUR MAX-SAT *et choisir la meilleure des deux solutions.*

Preuve. Nous utilisons les notations des preuves précédentes. L'algorithme renvoie un assignement, qui satisfait les clauses, de poids total au moins égal à

$$
\begin{aligned}
& \max\{\mathrm{Exp}(c(R)), \mathrm{Exp}(c(R_{y^*}))\} \\
\geq\ & \frac{1}{2}\left(\mathrm{Exp}(c(R)) + \mathrm{Exp}(c(R_{y^*}))\right) \\
\geq\ & \frac{1}{2}\sum_{j=1}^{m}\left(\left(1-2^{-|Z_j|}\right)c_j + \left(1-\left(1-\frac{1}{|Z_j|}\right)^{|Z_j|}\right)z_j^* c_j\right) \\
\geq\ & \frac{1}{2}\sum_{j=1}^{m}\left(2-2^{-|Z_j|}-\left(1-\frac{1}{|Z_j|}\right)^{|Z_j|}\right)z_j^* c_j \\
\geq\ & \frac{3}{4}\sum_{j=1}^{m} z_j^* c_j .
\end{aligned}
$$

Pour la dernière inégalité, observons que $2-2^{-k}-\left(1-\frac{1}{k}\right)^k \geq \frac{3}{2}$ pour tout $k \in \mathbb{N}$: pour $k \in \{1,2\}$ on a égalité ; pour $k \geq 3$ on a $2-2^{-k}-\left(1-\frac{1}{k}\right)^k \geq 2-\frac{1}{8}-\frac{1}{e} > \frac{3}{2}$. Comme l'optimum est au moins égal à $\sum_{j=1}^{m} z_j^* c_j$, le théorème est prouvé. □

Des algorithmes d'approximation légèrement meilleurs pour MAX-SAT (utilisant la programmation semi-définie) ont été trouvés ; voir Goemans et Williamson [1995], Mahajan et Ramesh [1999], et Feige et Goemans [1995]. Le meilleur algorithme connu actuellement atteint un rapport de performance égal à $1,270$ (Asano [2006]).

En fait, Bellare et Sudan [1994] ont montré que l'approximation de MAX-SAT à un facteur $\frac{74}{73}$ près est *NP*-difficile. Même pour MAX-3SAT (qui est le problème MAX-SAT restreint aux instances où chaque clause contient exactement trois littéraux) il n'existe pas de schéma d'approximation (à moins que $P = NP$), comme nous le verrons au paragraphe suivant.

16.6 Théorème *PCP*

De nombreux résultats de non-approximabilité sont fondés sur un théorème profond qui donne une nouvelle caractérisation de la classe *NP*. Rappelons qu'un problème de décision appartient à *NP* si et seulement s'il existe un algorithme de vérification de certificat polynomial. Nous considérons maintenant des algorithmes de vérification de certificat randomisés qui lisent l'instance complètement, mais seulement une petite partie du certificat devant être vérifié. Ils acceptent toujours des instances-«oui» avec des certificats corrects, mais parfois aussi des instances-«non».

On décide à l'avance et aléatoirement quels sont les bits du certificat qui sont lus ; plus précisément cette décision dépend de l'instance x et de $O(\log(\text{taille}(x)))$ bits aléatoires.

Nous formalisons maintenant ce concept. Si s est une chaîne et $t \in \mathbb{N}^k$, alors s_t désigne la chaîne de longueur k dont la i-ième composante est la t_i-ième composante de s ($i = 1, \ldots, k$).

Définition 16.31. *Un problème de décision (X, Y) est dans la classe* ***PCP*****(log *n*,1)** *s'il existe un polynôme p et une constante $k \in \mathbb{N}$, une fonction*

$$f : \left\{(x, r) : x \in X,\ r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x)))\rfloor}\right\} \to \mathbb{N}^k$$

calculable en temps polynomial, avec $f(x, r) \in \{1, \ldots, \lfloor p(\text{taille}(x))\rfloor\}^k$ pour tout x et r, et un problème de décision (X', Y') dans P, où $X' := \{(x, \pi, \gamma) : x \in X,\ \pi \in \{1, \ldots, \lfloor p(\text{taille}(x))\rfloor\}^k,\ \gamma \in \{0,1\}^k\}$, tels que, pour toute instance $x \in X$:
Si $x \in Y$, il existe un $c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor}$ tel que $\text{Prob}\left((x, f(x, r), c_{f(x,r)}) \in Y'\right) = 1$. *Si $x \notin Y$, alors* $\text{Prob}\left((x, f(x, r), c_{f(x,r)}) \in Y'\right) < \frac{1}{2}$ *pour tout $c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor}$.*

La probabilité est ici définie par rapport à une distribution uniforme des chaînes aléatoires $r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x)))\rfloor}$.

Le terme «*PCP*» correspond ainsi à une preuve vérifiable en probabilité (en anglais *probabilistically checkable proof*). Les paramètres $\log n$ et 1 reflètent que, pour une instance de taille n, $O(\log n)$ bits aléatoires sont utilisés et $O(1)$ bits du certificat sont lus.

Pour toute instance-«oui», il existe un certificat qui est toujours accepté ; alors que pour les instances-«non», il n'y a pas de chaîne qui soit acceptée en tant que certificat avec une probabilité supérieure ou égale à $\frac{1}{2}$. Remarquons que cette probabilité d'erreur égale à $\frac{1}{2}$ peut être remplacée de manière équivalente par tout nombre compris entre zéro et un (exercice 19).

Proposition 16.32. *PCP*$(\log n, 1) \subseteq$ *NP*.

Preuve. Soit $(X, Y) \in$ *PCP*$(\log n, 1)$, et soient $p, k, f, (X', Y')$ donnés comme dans la définition 16.31. Soit $X'' := \left\{(x, c) : x \in X,\ c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor}\right\}$, et

$$Y'' := \left\{(x, c) \in X'' : \text{Prob}\left((x, f(x, r), c_{f(x,r)}) \in Y'\right) = 1\right\}.$$

Pour montrer que $(X, Y) \in$ *NP*, il suffit de montrer que $(X'', Y'') \in P$. Mais puisqu'il existe seulement $2^{\lfloor \log(p(\text{taille}(x)))\rfloor}$, c.-à-d. au plus $p(\text{taille}(x))$ chaînes $r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x)))\rfloor}$, on peut toutes les essayer. Pour chacune d'entre elles on calcule $f(x, r)$ et on teste si $(x, f(x, r), c_{f(x,r)}) \in Y'$ (on utilise le fait que $(X', Y') \in P$). Le temps de calcul global est polynomial par rapport à $\text{taille}(x)$. □

Le résultat surprenant est alors que ces vérificateurs randomisés, qui lisent seulement un nombre constant de bits du certificat, sont aussi efficaces que les algorithmes standards (déterministes) de vérification de certificat qui ont toute l'information. Cela est appelé le théorème *PCP* :

Théorème 16.33. (Arora *et al.* [1998])

$$NP = PCP(\log n, 1).$$

La preuve de *NP* $\subseteq$ *PCP*$(\log n, 1)$ est très difficile et au-delà de l'étendue de ce livre. Elle est fondée sur des résultats obtenus plus tôt (et plus faibles) de Feige *et al.* [1996] et Arora et Safra [1998]. Pour une preuve exhaustive du théorème *PCP* 16.33, voir également Arora [1994], Hougardy, Prömel et Steger [1994], ou Ausiello *et al.* [1999]. Des résultats plus forts ont été trouvés successivement par Bellare, Goldreich et Sudan [1998] et Håstad [2001]. Par exemple, le nombre k de la définition 16.31 peut être choisi égal à 9. Une nouvelle preuve du théorème *PCP* a été proposée par Dinur [2007].

Nous montrons maintenant quelques-unes de ses conséquences pour la non-approximabilité de problèmes d'optimisation combinatoire. Nous commençons avec le PROBLÈME DE LA CLIQUE MAXIMUM et le PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM : étant donné un graphe non orienté G, trouver une clique, ou un ensemble stable, de cardinalité maximum dans G.

Rappelons la proposition 2.2 (et le corollaire 15.24) : les problèmes qui consistent à trouver une clique maximum, un ensemble stable maximum, ou une couverture minimum par les sommets sont tous équivalents. Cependant, l'algorithme d'approximation de facteur 2 pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS (paragraphe 16.1) n'implique pas un algorithme d'approximation pour le PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM ou pour le PROBLÈME DE LA CLIQUE MAXIMUM.

En fait, il peut arriver que l'algorithme renvoie une couverture par les sommets C de taille $n-2$, alors que l'optimum est égal à $\frac{n}{2}-1$ (où $n = |V(G)|$). Le complémentaire $V(G) \setminus C$ est alors un ensemble stable de cardinalité 2, mais l'ensemble stable maximum est de cardinalité $\frac{n}{2}+1$. Cet exemple montre qu'appliquer un algorithme à un autre problème à l'aide d'une transformation polynomiale ne préserve généralement pas sa garantie de performance. Nous considérerons un type de transformation particulier dans le paragraphe suivant. Nous déduisons ici un résultat de non-approximabilité pour le PROBLÈME DE LA CLIQUE MAXIMUM à partir du théorème *PCP* :

Théorème 16.34. (Arora et Safra [1998]) *À moins que P = NP, il n'existe pas d'algorithme d'approximation de facteur* 2 *pour le* PROBLÈME DE LA CLIQUE MAXIMUM.

Preuve. Soit $\mathcal{P} = (X, Y)$ un problème *NP*-complet. D'après le théorème *PCP* 16.33, $\mathcal{P} \in$ *PCP*$(\log n, 1)$, considérons donc p, k, f, $\mathcal{P}' := (X', Y')$ comme à la définition 16.31.

Pour tout $x \in X$ donné, on construit un graphe G_x de la façon suivante. Soit

$$V(G_x) := \left\{(r,a) : r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x))) \rfloor},\ a \in \{0,1\}^k,\ (x, f(x,r), a) \in Y'\right\}$$

(qui représente toutes les «exécutions acceptables» de l'algorithme randomisé de vérification de certificat). Deux sommets (r,a) et (r',a') sont reliés par une arête si $a_i = a'_j$ dès que la i-ième composante de $f(x,r)$ est égale à la j-ième composante de $f(x,r')$. Puisque $\mathcal{P}' \in P$ et qu'il existe seulement un nombre polynomial

de chaînes aléatoires, G_x peut être calculé en temps polynomial (et est de taille polynomiale).

Si $x \in Y$, alors, par définition, il existe un certificat $c \in \{0,1\}^{\lfloor p(\text{taille}(x))\rfloor}$ tel que $(x, f(x,r), c_{f(x,r)}) \in Y'$ pour tout $r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x)))\rfloor}$. Ainsi il existe une clique de taille $2^{\lfloor \log(p(\text{taille}(x)))\rfloor}$ dans G_x.

D'autre part, si $x \notin Y$, alors il n'existe pas de clique de taille $\frac{1}{2}2^{\lfloor \log(p(\text{taille}(x)))\rfloor}$ dans G_x : supposons que $(r^{(1)}, a^{(1)}), \ldots, (r^{(t)}, a^{(t)})$ soient les sommets d'une clique. Alors $r^{(1)}, \ldots, r^{(t)}$ sont deux à deux différents. On pose $c_i := a_k^{(j)}$ dès que la k-ième composante de $f(x, r^{(j)})$ est égale à i, et on fixe arbitrairement les composantes restantes de c (s'il en reste). De cette façon, on obtient un certificat c tel que $(x, f(x, r^{(i)}), c_{f(x,r^{(i)})}) \in Y'$ pour tout $i = 1, \ldots, t$. Si $x \notin Y$, on a $t < \frac{1}{2}2^{\lfloor \log(p(\text{taille}(x)))\rfloor}$.

Donc tout algorithme d'approximation de facteur 2 pour le PROBLÈME DE LA CLIQUE MAXIMUM permet de décider si $x \in Y$, c.-à-d. de résoudre $\mathcal{P}$. Puisque $\mathcal{P}$ est *NP*-complet, cela n'est possible que si $P = NP$. □

La réduction dans la preuve précédente est due à Feige *et al.* [1996]. Puisque la probabilité d'erreur, égale à $\frac{1}{2}$ dans la définition 16.31, peut être remplacée par tout nombre compris entre 0 et 1 (exercice 19), on obtient qu'il n'existe pas d'algorithme d'approximation de facteur ρ pour le PROBLÈME DE LA CLIQUE MAXIMUM quel que soit $\rho \geq 1$ (à moins que $P = NP$).

En allant un peu plus loin Zuckerman [2006] a montré que, à moins que $P = NP$, il n'existe pas d'algorithme polynomial calculant la taille maximum d'une clique dans un graphe sur n sommets à un facteur $n^{1-\epsilon}$ près, pour tout $\epsilon > 0$ fixé. Le meilleur algorithme connu garantit de trouver une clique de taille $\frac{k \log^3 n}{n(\log\log n)^2}$ dans ce cas (Feige [2004]). Bien entendu, tout cela est aussi vérifié pour le PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM (en considérant le complémentaire du graphe donné).

Nous passons maintenant à la restriction suivante de MAX-SAT :

MAX-3SAT	
Instance	Un ensemble X de variables et une famille $\mathcal{Z}$ de clauses sur X, chacune contenant exactement trois littéraux.
Tâche	Trouver un assignement T de X tel que le nombre de clauses de $\mathcal{Z}$ qui sont satisfaites par T soit maximum.

Au paragraphe 16.5, nous avions présenté un algorithme d'approximation simple de facteur $\frac{8}{7}$ pour MAX-3SAT, même pour le cas pondéré (théorème 16.28). Håstad [2001] a montré que l'on ne peut faire mieux : il ne peut exister aucun algorithme d'approximation pour MAX-3SAT de facteur ρ avec $\rho < \frac{8}{7}$ à moins que $P = NP$. Nous prouvons ici le résultat plus faible suivant :

Théorème 16.35. (Arora *et al.* [1998]) *À moins que $P = NP$, il n'existe aucun schéma d'approximation pour* MAX-3SAT.

Preuve. Soit $\mathcal{P} = (X, Y)$ un problème *NP*-complet. D'après le théorème *PCP* 16.33, $\mathcal{P} \in PCP(\log n, 1)$, considérons donc p, k, f, $\mathcal{P}' := (X', Y')$ comme dans la définition 16.31.

Pour tout $x \in X$ donné, on construit une instance J_x du problème 3SAT. Autrement dit, pour chaque chaîne aléatoire $r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x))) \rfloor}$ on définit une famille $\mathcal{Z}_r$ de 3SAT-clauses (l'union de toutes ces familles sera J_x). Nous construisons d'abord une famille $\mathcal{Z}'_r$ de clauses ayant un nombre arbitraire de littéraux et appliquons ensuite la proposition 15.21.

Considérons donc $r \in \{0,1\}^{\lfloor \log(p(\text{taille}(x))) \rfloor}$ et $f(x,r) = (t_1, \ldots, t_k)$. Soit $\{a^{(1)}, \ldots, a^{(s_r)}\}$ l'ensemble des chaînes $a \in \{0,1\}^k$ telles que $(x, f(x,r), a) \in Y'$. Si $s_r = 0$ alors on pose simplement $\mathcal{Z}' := \{\{y\}, \{\bar{y}\}\}$, où y est une variable qui n'est pas déjà utilisée ailleurs.

Sinon soit $c \in \{0,1\}^{\lfloor p(\text{taille}(x)) \rfloor}$. On a que $(x, f(x,r), c_{f(x,r)}) \in Y'$ si et seulement si

$$\bigvee_{j=1}^{s_r} \left(\bigwedge_{i=1}^{k} \left(c_{t_i} = a_i^{(j)} \right) \right).$$

Cela est équivalent à

$$\bigwedge_{(i_1,\ldots,i_{s_r}) \in \{1,\ldots,k\}^{s_r}} \left(\bigvee_{j=1}^{s_r} \left(c_{t_{i_j}} = a_i^{(j)} \right) \right).$$

Cette conjonction de clauses peut être construite en temps polynomial, car $\mathcal{P}' \in P$ et k est une constante. En prenant des variables booléennes $\pi_1, \ldots, \pi_{\lfloor p(\text{taille}(x)) \rfloor}$ représentant les bits $c_1, \ldots, c_{\lfloor p(\text{taille}(x)) \rfloor}$, on obtient une famille $\mathcal{Z}'_r$ de k^{s_r} clauses (chacune de ces clauses contenant s_r littéraux) telle que $\mathcal{Z}'_r$ est satisfaite si et seulement si $(x, f(x,r), c_{f(x,r)}) \in Y'$.

D'après la proposition 15.21, on peut réécrire chaque famille $\mathcal{Z}'_r$, de manière équivalente, comme une conjonction de 3SAT-clauses, où le nombre de clauses augmente au plus d'un facteur $\max\{s_r - 2, 4\}$. Soit $\mathcal{Z}_r$ cette famille de clauses. Puisque $s_r \leq 2^k$, chaque famille $\mathcal{Z}_r$ est constituée d'au plus $l := k^{2^k} \max\{2^k - 2, 4\}$ 3SAT-clauses.

Notre intance J_x du problème 3SAT est l'union de toutes les familles $\mathcal{Z}_r$ pour tout r. J_x peut être calculé en temps polynomial.

Si x est maintenant une instance-«oui», alors il existe un certificat c comme à la définition 16.31. Ce certificat c définit de manière immédiate un assignement satisfaisant J_x.

D'autre part, si x est une instance-«non», alors seulement la moitié des formules $\mathcal{Z}_r$ sont simultanément satisfaisables. Donc, dans ce cas, tout assignement laisse au moins une fraction $\frac{1}{2l}$ des clauses insatisfaites.

Aussi tout algorithme d'approximation de facteur k pour MAX-3SAT tel que $k < \frac{2l}{2l-1}$ satisfait plus qu'une fraction $\frac{2l-1}{2l} = 1 - \frac{1}{2l}$ des clauses d'une instance satisfaisable. Ainsi un tel algorithme permet de décider si $x \in Y$ ou non. Puisque $\mathcal{P}$ est *NP*-complet, un tel algorithme ne peut exister à moins que $P = NP$. □

16.7 L-réductions

Notre objectif est de montrer, pour d'autres problèmes que MAX-3SAT, qu'ils n'ont pas de schémas d'approximation à moins que *P = NP*. Comme pour les preuves de *NP*-complétude (paragraphe 15.5), il n'est pas nécessaire d'avoir une preuve directe utilisant la définition de $PCP(\log n, 1)$ pour chaque problème. À la place, on utilise un type particulier de réduction qui préserve l'approximabilité (contrairement aux transformations polynomiales générales) :

Définition 16.36. *Considérons deux problèmes d'optimisation avec des poids non négatifs,* $\mathcal{P} = (X, (S_x)_{x\in X}, c, \text{but})$ *et* $\mathcal{P}' = (X', (S'_x)_{x\in X'}, c', \text{but}')$. *Une* **L-réduction** *de* $\mathcal{P}$ *à* $\mathcal{P}'$ *est une paire de fonctions* f *et* g*, toutes deux calculables en temps polynomial, et deux constantes* $\alpha, \beta > 0$ *telles que pour toute instance* x *de* $\mathcal{P}$ *:*

(a) $f(x)$ *est une instance de* $\mathcal{P}'$ *telle que* $\mathrm{OPT}(f(x)) \le \alpha\,\mathrm{OPT}(x)$.

(b) *Pour toute solution réalisable* y' *de* $f(x)$, $g(x, y')$ *est une solution réalisable de* x *telle que* $|c(x, g(x, y')) - \mathrm{OPT}(x)| \le \beta|c'(f(x), y') - \mathrm{OPT}(f(x))|$.

On dit que $\mathcal{P}$ *est* **L-réductible** *à* $\mathcal{P}'$ *s'il existe une L-réduction de* $\mathcal{P}$ *à* $\mathcal{P}'$.

La lettre «L» dans l'expression L-réduction correspond à «linéaire». Les L-réductions ont été introduites par Papadimitriou et Yannakakis [1991]. La définition implique de manière immédiate que les L-réductions peuvent être composées :

Proposition 16.37. *Soient* $\mathcal{P}$, $\mathcal{P}'$, $\mathcal{P}''$ *des problèmes d'optimisation avec des poids non négatifs. Si* (f, g, α, β) *est une L-réduction de* $\mathcal{P}$ *à* $\mathcal{P}'$ *et* $(f', g', \alpha', \beta')$ *est une L-réduction de* $\mathcal{P}'$ *à* $\mathcal{P}''$*, alors leur composée* $(f'', g'', \alpha\alpha', \beta\beta')$ *est une L-réduction de* $\mathcal{P}$ *à* $\mathcal{P}''$*, où* $f''(x) = f'(f(x))$ *et* $g''(x, y'') = g(x, g'(x', y''))$. □

La propriété décisive des L-réductions est qu'elles préservent l'approximabilité :

Théorème 16.38. (Papadimitriou et Yannakakis [1991]) *Soient* $\mathcal{P}$ *et* $\mathcal{P}'$ *deux problèmes d'optimisation avec des poids non négatifs. Soit* (f, g, α, β) *une L-réduction de* $\mathcal{P}$ *à* $\mathcal{P}'$*. S'il existe un schéma d'approximation pour* $\mathcal{P}'$*, alors il existe un schéma d'approximation pour* $\mathcal{P}$.

Preuve. Étant donné une instance x de $\mathcal{P}$ et un nombre $0 < \epsilon < 1$, on applique le schéma d'approximation pour $\mathcal{P}'$ à $f(x)$ et $\epsilon' := \frac{\epsilon}{2\alpha\beta}$. On obtient une solution réalisable y' de $f(x)$ et on retourne finalement $y := g(x, y')$, une solution réalisable de x. Comme

$$\begin{aligned}
|c(x,y) - \mathrm{OPT}(x)| &\le \beta|c'(f(x),y') - \mathrm{OPT}(f(x))| \\
&\le \beta \max\Big\{(1+\epsilon')\,\mathrm{OPT}(f(x)) - \mathrm{OPT}(f(x)), \\
&\qquad\qquad \mathrm{OPT}(f(x)) - \frac{1}{1+\epsilon'}\,\mathrm{OPT}(f(x))\Big\} \\
&\le \beta\epsilon'\,\mathrm{OPT}(f(x)) \\
&\le \alpha\beta\epsilon'\,\mathrm{OPT}(x) \\
&= \frac{\epsilon}{2}\,\mathrm{OPT}(x)
\end{aligned}$$

on obtient

$$c(x,y) \;\leq\; \mathrm{OPT}(x) + |c(x,y) - \mathrm{OPT}(x)| \;\leq\; \left(1 + \frac{\epsilon}{2}\right) \mathrm{OPT}(x)$$

et

$$c(x,y) \;\geq\; \mathrm{OPT}(x) - |\,\mathrm{OPT}(x) - c(x,y)| \;\geq\; \left(1 - \frac{\epsilon}{2}\right) \mathrm{OPT}(x) \;>\; \frac{1}{1+\epsilon}\,\mathrm{OPT}(x),$$

cela constitue donc un schéma d'approximation pour $\mathcal{P}$. □

Ce théorème ainsi que le théorème 16.35 motive la définition suivante :

Définition 16.39. *Un problème d'optimisation $\mathcal{P}$ avec des poids non négatifs est dit* ***MAXSNP*-difficile** *si* MAX-3SAT *est L-réductible à $\mathcal{P}$.*

Le terme *MAXSNP* se réfère à une classe de problèmes d'optimisation introduits par Papadimitriou et Yannakakis [1991]. Nous n'avons pas besoin ici de cette classe ; nous omettons donc sa définition (non triviale).

Corollaire 16.40. *À moins que P = NP, il n'existe pas de schéma d'approximation pour tout problème MAXSNP-difficile.*

Preuve. On obtient directement le résultat d'après les théorèmes 16.35 et 16.38. □

Nous prouverons que plusieurs problèmes sont *MAXSNP*-difficiles en décrivant des L-réductions. Nous commençons avec une version restreinte de MAX-3SAT :

PROBLÈME MAX-SAT 3-OCCURRENCES	
Instance	Un ensemble X de variables et une famille $\mathcal{Z}$ de clauses sur X, chacune contenant au plus trois littéraux, telle qu'aucune variable n'apparaisse dans plus de trois clauses.
Tâche	Trouver un assignement T de X tel que le nombre de clauses de $\mathcal{Z}$ qui sont satisfaites par T soit maximum.

Le fait que ce problème soit *NP*-difficile peut être prouvé par une simple transformation de 3SAT (ou MAX-3SAT) ; voir l'exercice 9 du chapitre 15. Puisque cette transformation n'est pas une L-réduction, cela n'implique pas que le problème soit *MAXSNP*-difficile. Nous avons besoin d'une construction plus compliquée, utilisant ce que l'on appelle les graphes expandeurs :

Définition 16.41. *Soient G un graphe non orienté et une constante $\gamma > 0$. G est un* **γ-expandeur** *si pour chaque $A \subseteq V(G)$ tel que $|A| \leq \frac{|V(G)|}{2}$ on a $|\Gamma(A)| \geq \gamma|A|$.*

Par exemple, un graphe complet est un 1-expandeur. Cependant, on s'intéresse aux expandeurs qui ont un petit nombre d'arêtes. Nous citons le théorème suivant sans donner sa preuve qui est relativement compliquée :

Théorème 16.42. (Ajtai [1994]) *Il existe une constante positive γ telle que, pour tout entier pair donné $n \geq 4$, un γ-expandeur 3-régulier sur n sommets peut être construit en temps $O(n^3 \log^3 n)$.*

Le corollaire suivant a été mentionné (et utilisé) par Papadimitriou [1994], et une preuve correcte a été donnée par Fernández-Baca et Lagergren [1998] :

Corollaire 16.43. *Pour tout entier $n \geq 3$ donné, un graphe orienté G sur $O(n)$ sommets et un ensemble $S \subseteq V(G)$ de cardinalité n avec les propriétés suivantes peuvent être construits en temps $O(n^3 \log^3 n)$:*
$|\delta^-(v)| + |\delta^+(v)| \leq 3$ *pour tout* $v \in V(G)$ *;*
$|\delta^-(v)| + |\delta^+(v)| = 2$ *pour tout* $v \in S$ *; et*
$|\delta^+(A)| \geq \min\{|S \cap A|, |S \setminus A|\}$ *pour tout* $A \subseteq V(G)$.

Preuve. Soit $\gamma > 0$ la constante du théorème 16.42, et considérons $k := \left\lceil \frac{1}{\gamma} \right\rceil$. Nous construisons d'abord un γ-expandeur 3-régulier H sur n ou $n+1$ sommets, en utilisant le théorème 16.42.

Nous remplaçons chaque arête (v, w) par k arcs parallèles (v, w) et k arcs parallèles (w, v). Notons H' le graphe orienté résultant. Remarquons que pour tout $A \subseteq V(H')$ tel que $|A| \leq \frac{|V(H')|}{2}$ on a

$$|\delta^+_{H'}(A)| \;=\; k|\delta_H(A)| \;\geq\; k|\Gamma_H(A)| \;\geq\; k\gamma|A| \;\geq\; |A|.$$

De manière similaire, on a pour tout $A \subseteq V(H')$ tel que $|A| > \frac{|V(H')|}{2}$:

$$\begin{aligned} |\delta^+_{H'}(A)| \;=\; k|\delta_H(V(H') \setminus A)| \;&\geq\; k|\Gamma_H(V(H') \setminus A)| \\ &\geq\; k\gamma|V(H') \setminus A| \;\geq\; |V(H') \setminus A|. \end{aligned}$$

Donc, dans les deux cas, on a $|\delta^+_{H'}(A)| \geq \min\{|A|, |V(H') \setminus A|\}$.

On divise maintenant chaque sommet $v \in V(H')$ en $6k+1$ sommets $x_{v,i}$, $i = 0, \ldots, 6k$, de telle façon que chaque sommet excepté $x_{v,0}$ soit de degré 1. Pour chaque sommet $x_{v,i}$ nous ajoutons maintenant des sommets $w_{v,i,j}$ et $y_{v,i,j}$ ($j = 0, \ldots, 6k$) reliés par un chemin de longueur $12k+2$ avec les sommets $w_{v,i,0}, w_{v,i,1}, \ldots, w_{v,i,6k}, x_{v,i}, y_{v,i,0}, \ldots, y_{v,i,6k}$ dans cet ordre. Finalement nous ajoutons des arcs $(y_{v,i,j}, w_{v,j,i})$ pour tout $v \in V(H')$, pour tout $i \in \{0, \ldots, 6k\}$ et pour tout $j \in \{0, \ldots, 6k\} \setminus \{i\}$.

Tout cela réuni, nous avons un ensemble de sommets Z_v de cardinalité $(6k+1)(12k+3)$ pour chaque $v \in V(H')$. Le graphe résultant G a $|V(H')|(6k+1)(12k+3) = O(n)$ sommets, chacun de degré deux ou trois. Par construction, $G[Z_v]$ contient $\min\{|X_1|, |X_2|\}$ chemins sommet-disjoints de X_1 à X_2 pour toute paire de sous-ensembles disjoints X_1, X_2 de $\{x_{v,i} : i = 0, \ldots, 6k\}$.

On choisit pour S un sous-ensemble de n éléments de $\{x_{v,0} : v \in V(H')\}$; remarquons que chacun de ces sommets a un arc entrant et un arc sortant.

Il reste à prouver que $|\delta^+(A)| \geq \min\{|S \cap A|, |S \setminus A|\}$ pour tout $A \subseteq V(G)$. On prouve cela par induction sur $|\{v \in V(H') : \emptyset \neq A \cap Z_v \neq Z_v\}|$. Si ce nombre est égal à zéro, c.-à-d. si $A = \bigcup_{v \in B} Z_v$ pour un certain $B \subseteq V(H')$, alors on a

$$|\delta_G^+(A)| \;=\; |\delta_{H'}^+(B)| \;\geq\; \min\{|B|, |V(H') \setminus B|\} \;\geq\; \min\{|S \cap A|, |S \setminus A|\}.$$

Sinon soit $v \in V(H')$ tel que $\emptyset \neq A \cap Z_v \neq Z_v$. Soient $P := \{x_{v,i} : i = 0, \ldots, 6k\} \cap A$ et $Q := \{x_{v,i} : i = 0, \ldots, 6k\} \setminus A$. Si $|P| \leq 3k$, alors d'après la propriété de $G[Z_v]$ on a

$$\begin{aligned} |E_G^+(Z_v \cap A, Z_v \setminus A)| &\geq |P| \;=\; |P \setminus S| + |P \cap S| \\ &\geq |E_G^+(A \setminus Z_v, A \cap Z_v)| + |P \cap S|. \end{aligned}$$

En appliquant l'hypothèse d'induction à $A \setminus Z_v$, nous obtenons ainsi

$$\begin{aligned} |\delta_G^+(A)| &\geq |\delta_G^+(A \setminus Z_v)| + |P \cap S| \\ &\geq \min\{|S \cap (A \setminus Z_v)|, |S \setminus (A \setminus Z_v)|\} + |P \cap S| \\ &\geq \min\{|S \cap A|, |S \setminus A|\}. \end{aligned}$$

De manière similaire, si $|P| \geq 3k + 1$, alors $|Q| \leq 3k$ et d'après la propriété de $G[Z_v]$ on a

$$\begin{aligned} |E_G^+(Z_v \cap A, Z_v \setminus A)| &\geq |Q| \;=\; |Q \setminus S| + |Q \cap S| \\ &\geq |E_G^+(Z_v \setminus A, V(G) \setminus (A \cup Z_v))| + |Q \cap S|. \end{aligned}$$

En appliquant l'hypothèse d'induction à $A \cup Z_v$, on obtient donc

$$\begin{aligned} |\delta_G^+(A)| &\geq |\delta_G^+(A \cup Z_v)| + |Q \cap S| \\ &\geq \min\{|S \cap (A \cup Z_v)|, |S \setminus (A \cup Z_v)|\} + |Q \cap S| \\ &\geq \min\{|S \cap A|, |S \setminus A|\}. \end{aligned}$$

□

On peut maintenant prouver :

Théorème 16.44. (Papadimitriou et Yannakakis [1991], Papadimitriou [1994], Fernández-Baca et Lagergren [1998]) *Le* PROBLÈME MAX-SAT 3-OCCURRENCES *est MAXSNP-difficile.*

Preuve. Nous décrivons une L-réduction (f, g, α, β) à partir de MAX-3SAT. Pour définir f, considérons $(X, \mathcal{Z})$ une instance de MAX-3SAT. Pour chaque variable $x \in X$ qui apparaît dans plus que trois clauses, disons dans k clauses, nous modifions l'instance de la façon suivante. Nous remplaçons x par une nouvelle variable différente dans chaque clause. De cette façon, nous introduisons de nouvelles variables $x_1, \ldots, x_k$. Nous introduisons des variables et des contraintes supplémentaires qui assurent qu'il est favorable d'assigner la même valeur à toutes les variables $x_1, \ldots, x_k$.

Nous construisons G et S comme au corollaire 16.43 et renommons les sommets de telle façon que $S = \{1, \ldots, k\}$. Nous introduisons maintenant une nouvelle

variable x_v pour chaque sommet $v \in V(G) \setminus S$, et nous introduisons une clause $\{\overline{x_v}, x_w\}$ pour chaque arête $(v,w) \in E(G)$. Au total nous avons ajouté au plus

$$\frac{3}{2}(k+1)\left(6\left\lceil\frac{1}{\gamma}\right\rceil+1\right)\left(12\left\lceil\frac{1}{\gamma}\right\rceil+3\right) \;\leq\; 315\left\lceil\frac{1}{\gamma}\right\rceil^2 k$$

nouvelles clauses, où γ est encore la constante du théorème 16.42.

En appliquant la substitution précédente à chaque variable, on obtient une instance $(X', \mathcal{Z}') = f(X, \mathcal{Z})$ du PROBLÈME MAX-SAT 3-OCCURRENCES telle que

$$|\mathcal{Z}'| \;\leq\; |\mathcal{Z}| + 315\left\lceil\frac{1}{\gamma}\right\rceil^2 3|\mathcal{Z}| \;\leq\; 946\left\lceil\frac{1}{\gamma}\right\rceil^2 |\mathcal{Z}|.$$

Ainsi

$$\mathrm{OPT}(X', \mathcal{Z}') \;\leq\; |\mathcal{Z}'| \;\leq\; 946\left\lceil\frac{1}{\gamma}\right\rceil^2 |\mathcal{Z}| \;\leq\; 1892\left\lceil\frac{1}{\gamma}\right\rceil^2 \mathrm{OPT}(X, \mathcal{Z}),$$

car au moins la moitié des clauses d'une instance de MAX-SAT peuvent être satisfaites (en fixant soit toutes les variables à *vrai*, soit toutes à *faux*). On peut donc poser $\alpha := 1892\left\lceil\frac{1}{\gamma}\right\rceil^2$.

Pour décrire g, considérons un assignement T' de X'. Nous construisons d'abord un assignement de T'' de X' satisfaisant au moins autant de clauses de $\mathcal{Z}'$ que T', et satisfaisant toutes les nouvelles clauses (qui correspondent aux arcs du graphe G précédent). Pour toute variable x apparaissant plus que trois fois dans $(X, \mathcal{Z})$, considérons G le graphe construit précédemment, et soit $A := \{v \in V(G) : T'(x_v) = \textit{vrai}\}$. Si $|S \cap A| \geq |S \setminus A|$, alors on pose $T''(x_v) := \textit{vrai}$ pour tout $v \in V(G)$, sinon on pose $T''(x_v) := \textit{faux}$ pour tout $v \in V(G)$. Il est clair que toutes les nouvelles clauses (qui correspondent aux arcs) sont satisfaites.

Il existe au plus $\min\{|S \cap A|, |S \setminus A|\}$ anciennes clauses satisfaites par T', mais pas par T''. D'autre part, T' ne satisfait aucune des clauses $\{\overline{x_v}, x_w\}$ pour $(v,w) \in \delta_G^+(A)$. D'après les propriétés de G, le nombre de ces clauses est au moins égal à $\min\{|S \cap A|, |S \setminus A|\}$.

T'' fournit alors un assignement $T = g(X, \mathcal{Z}, T')$ de X de la manière évidente suivante. Posons $T(x) := T''(x) = T'(x)$ pour $x \in X \cap X'$ et $T(x) := T''(x_i)$ si x_i est une variable remplaçant x dans la construction de $(X, \mathcal{Z})$ à $(X', \mathcal{Z}')$.

T viole autant de clauses que T''. Donc si $c(X, \mathcal{Z}, T)$ désigne le nombre de clauses de l'instance $(X, \mathcal{Z})$ qui sont satisfaites par T, on conclut que

$$|\mathcal{Z}| - c(X, \mathcal{Z}, T) \;=\; |\mathcal{Z}'| - c(X', \mathcal{Z}', T'') \;\leq\; |\mathcal{Z}'| - c(X', \mathcal{Z}', T'). \qquad (16.7)$$

D'autre part, tout assignement de T de X conduit à un assignement T' de X' qui viole le même nombre de clauses (en affectant les variables x_v ($v \in V(G)$) uniformément à $T(x)$ pour chaque variable x et le graphe correspondant G de la construction précédente). Ainsi

$$|\mathcal{Z}| - \mathrm{OPT}(X, \mathcal{Z}) \;\geq\; |\mathcal{Z}'| - \mathrm{OPT}(X', \mathcal{Z}'). \tag{16.8}$$

En combinant (16.7) et (16.8) on obtient

$$\begin{aligned} |\,\mathrm{OPT}(X, \mathcal{Z}) - c(X, \mathcal{Z}, T)| &= (|\mathcal{Z}| - c(X, \mathcal{Z}, T)) - (|\mathcal{Z}| - \mathrm{OPT}(X, \mathcal{Z})) \\ &\leq \mathrm{OPT}(X', \mathcal{Z}') - c(X', \mathcal{Z}', T') \\ &= |\,\mathrm{OPT}(X', \mathcal{Z}') - c(X', \mathcal{Z}', T')|, \end{aligned}$$

où $T = g(X, \mathcal{Z}, T')$. Donc $(f, g, \alpha, 1)$ est bien une L-réduction. □

Ce résultat est le point de départ, pour plusieurs problèmes, de preuves établissant qu'ils sont *MAXSNP*-difficiles. Par exemple :

Corollaire 16.45. (Papadimitriou et Yannakakis [1991]) *Le* PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM *restreint aux graphes de degré maximum égal à 4 est MAXSNP-difficile.*

Preuve. La construction de la preuve du théorème 15.23 définit une L-réduction du PROBLÈME MAX-SAT 3-OCCURRENCES au PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM restreint aux graphes de degré maximum égal à 4 : pour chaque instance $(X, \mathcal{Z})$ un graphe G est construit de telle façon que, de chaque assignement satisfaisant k clauses, on obtienne facilement un ensemble stable de cardinalité k, et vice versa. □

En effet, le PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM est *MAXSNP*-difficile même lorsqu'il est restreint aux graphes 3-réguliers (Berman et Fujito [1999]). D'autre part, un simple algorithme glouton, qui à chaque étape choisit un sommet v de degré minimum et supprime v et tous ses voisins, constitue un algorithme d'approximation de facteur $\frac{(k+2)}{3}$ pour le PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM dans les graphes de degré maximum égal à k (Halldórsson et Radhakrishnan [1997]). Pour $k = 4$, cela donne un rapport de performance égal à 2, ce qui est meilleur que le rapport égal à 8 que nous obtenons de la preuve suivante (qui utilise l'algorithme d'approximation de facteur 2 pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS).

Théorème 16.46. (Papadimitriou et Yannakakis [1991]) *Le* PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS *restreint aux graphes de degré maximum égal à 4 est MAXSNP-difficile.*

Preuve. Considérons la transformation triviale du PROBLÈME DE L'ENSEMBLE STABLE MAXIMUM (proposition 2.2) avec $f(G) := G$ et $g(G, X) := V(G) \setminus X$ pour tout graphe G et pour tout $X \subseteq V(G)$. Bien que ce ne soit pas en général une L-réduction, il s'agit bien d'une L-réduction si on se restreint aux graphes de degré maximum égal à 4, comme nous allons le montrer.

Si G est de degré maximum égal à 4, il existe un ensemble stable de cardinalité au moins égale à $\frac{|V(G)|}{5}$. Donc si on désigne par $\alpha(G)$ la cardinalité maximum

d'un ensemble stable et par $\tau(G)$ la cardinalité minimum d'une couverture par les sommets, on a

$$\alpha(G) \geq \frac{1}{4}(|V(G)| - \alpha(G)) = \frac{1}{4}\tau(G)$$

et $\alpha(G) - |X| = |V(G) \setminus X| - \tau(G)$ pour tout ensemble stable $X \subseteq V(G)$. Ainsi $(f, g, 4, 1)$ est une L-réduction. □

Voir Clementi et Trevisan [1999] et Chlebík et Chlebíková [2006] pour des résultats plus poussés. En particulier, il n'existe pas de schéma d'approximation pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS (à moins que *P = NP*). Nous prouverons que d'autres problèmes sont *MAXSNP*-difficiles aux chapitres suivants ; voir également l'exercice 22.

Exercices

1. Formuler un algorithme d'approximation de facteur 2 pour le problème suivant. Étant donné un graphe orienté avec des poids sur les arcs, trouver un sous-graphe orienté sans circuit de poids maximum.
   Remarque : on ne connaît pas d'algorithme d'approximation de facteur k avec $k < 2$ pour ce problème.
2. Le PROBLÈME DU k-CENTRE est défini de la manière suivante : étant donné un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$, et un nombre $k \in \mathbb{N}$, trouver un ensemble $X \subseteq V(G)$ de cardinalité k tel que

   $$\max_{v \in V(G)} \min_{x \in X} \mathrm{dist}(v, x)$$

   soit minimum. On désignera la valeur optimale par $\mathrm{OPT}(G, c, k)$.
   (a) Soit S un ensemble stable maximum dans $(V(G), \{(v, w) : \mathrm{dist}(v, w) \leq 2R\})$. Montrer qu'alors $\mathrm{OPT}(G, c, |S| - 1) > R$.
   (b) Utiliser (a) pour décrire un algorithme d'approximation de facteur 2 pour le PROBLÈME DU k-CENTRE.
   (Hochbaum et Shmoys [1985])

∗ (c) Montrer qu'il n'existe pas d'algorithme d'approximation de facteur r pour le PROBLÈME DU k-CENTRE pour tout $r < 2$.
   Indication : utiliser l'exercice 12 du chapitre 15.
   (Hsu et Nemhauser [1979])

3. Peut-on trouver une couverture minimum par les sommets (ou un ensemble stable maximum) dans un graphe biparti en temps polynomial ?
4. Montrer que la garantie de performance dans le théorème 16.5 est atteinte.
5. Montrer que l'algorithme suivant est un algorithme d'approximation de facteur 2 pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS. Calculer un arbre-DFS et fournir tous ses sommets de degré sortant non nul.
   (Bar-Yehuda [non publié])

6. Montrer que la relaxation linéaire $\min\{cx : M^\top x \geq \mathbb{1},\ x \geq 0\}$ du PROBLÈME DE LA COUVERTURE PAR LES SOMMETS DE POIDS MINIMUM, où M est la matrice d'incidence d'un graphe non orienté et $c \in \mathbb{R}_+^{V(G)}$, a toujours une solution optimale demi-entière (c.-à-d. une solution dont les entrées sont seulement égales à $0, \frac{1}{2}, 1$). Déduire de ce résultat un autre algorithme d'approximation de facteur 2.

* 7. Considérons le PROBLÈME DU SOMMET-TRANSVERSAL DES CYCLES DE POIDS MINIMUM : étant donné un graphe non orienté G et des poids $c : V(G) \to \mathbb{R}_+$, trouver un ensemble de sommets $X \subseteq V(G)$ de poids minimum tel que $G - X$ soit une forêt. Considérons l'algorithme récursif A suivant : si $E(G) = \emptyset$, alors retourner $A(G, c) := \emptyset$. Si $|\delta_G(x)| \leq 1$ pour un certain $x \in V(G)$, alors retourner $A(G, c) := A(G - x, c)$. Si $c(x) = 0$ pour un certain $x \in V(G)$, alors retourner $A(G, c) := \{x\} \cup A(G - x, c)$. Sinon soient

$$\epsilon := \min_{x \in V(G)} \frac{c(v)}{|\delta(v)|}$$

et $c'(v) := c(v) - \epsilon|\delta(v)|$ ($v \in V(G)$). Soit $X := A(G, c')$. Pour chaque $x \in X$ faire : si $G - (X \setminus \{x\})$ est une forêt, alors poser $X := X \setminus \{x\}$. Retourner $A(G, c) := x$.

Prouver qu'il s'agit d'un algorithme d'approximation de facteur 2 pour le problème du transversal des cycles par les sommets de poids minimum.
(Becker et Geiger [1996])

8. Montrer que le PROBLÈME DE LA COUPE MAXIMUM est *NP*-difficile même pour les graphes simples.

9. Prouver que le simple algorithme glouton pour MAX-CUT décrit au début du paragraphe 16.2 est un algorithme d'approximation de facteur 2.

10. Considérons l'algorithme de recherche locale suivant pour le PROBLÈME DE LA COUPE MAXIMUM. Partir d'un sous-ensemble propre non vide S de $V(G)$. Ensuite vérifier itérativement si un sommet peut être ajouté à S ou supprimé de S de telle façon que $|\delta(S)|$ augmente. Arrêter si aucune amélioration de ce type n'est possible.

    (a) Prouver que l'algorithme précédent est un algorithme d'approximation de facteur 2. (Se reporter à l'exercice 10 du chapitre 2.)

    (b) L'algorithme peut-il être étendu au PROBLÈME DE LA COUPE DE POIDS MAXIMUM, où on a des poids non négatifs sur les arêtes ?

    (c) L'algorithme précédent trouve-t-il une solution optimale pour les graphes planaires, ou pour les graphes bipartis ? Pour ces deux classes de graphes il existe un algorithme polynomial (exercice 7 du chapitre 12 et proposition 2.27).

11. Considérons le PROBLÈME DE LA COUPE SORTANTE DE POIDS MAXIMUM : étant donné un graphe orienté G avec des poids $c : E(G) \to \mathbb{R}_+$, on recherche un ensemble $X \subseteq V(G)$ tel que $\sum_{e \in \delta^+(X)} c(e)$ soit maximum. Montrer qu'il

existe un algorithme d'approximation de facteur 4 pour ce problème.
Indication : utiliser l'exercice 10.
Remarque : il existe un algorithme d'approximation de facteur 1,165 mais pas de facteur 1,09 à moins que $P = NP$ (Feige et Goemans [1995], Håstad [2001]).

12. Montrer que $(\frac{1}{\pi}\arccos(y_i^\top y_j)_{1\le i,j\le n}$ est une combinaison convexe de coupes semi-métriques δ^R, $R \subseteq \{1,\dots,n\}$, où $\delta^R_{i,j} = 1$ si $|R \cap \{i,j\}| = 1$ et $\delta^R_{i,j} = 0$ sinon.
Indication : écrire

$$(\mu(H(y_i)\triangle H(y_j))_{1\le i,j\le n} = \sum_{R\subseteq\{1,\dots,n\}} \mu\left(\bigcap_{i\in R} H(y_i) \setminus \bigcup_{i\notin R} H(y_i)\right)\delta^R.$$

Remarque : voir Deza et Laurent [1997] pour de nombreuses informations connexes.

13. Montrer que, pour tout $n \in \mathbb{N}$, il existe un graphe biparti sur $2n$ sommets pour lequel l'ALGORITHME GLOUTON DE COLORATION nécessite n couleurs. Ainsi l'algorithme peut fournir de très mauvais résultats. Cependant, montrer qu'il existe toujours un ordre sur les sommets pour lequel l'algorithme trouve une coloration optimale.

14. Montrer que l'on peut colorer tout graphe 3-colorable G à l'aide d'au plus $2\sqrt{2n}$ couleurs en temps polynomial, où $n := |V(G)|$.
Indication : tant qu'il existe un sommet v de degré au moins égal à $\sqrt{2n}$, colorer $\Gamma(v)$ optimalement à l'aide d'au plus deux couleurs (qui ne seront pas utilisées ensuite), et supprimer ces sommets. Finalement, utiliser l'ALGORITHME GLOUTON DE COLORATION.
(Wigderson [1983])

15. Montrer que les graphes suivants sont parfaits :
    (a) Les graphes bipartis.
    (b) Les graphes d'intervalles : $(\{v_1,\dots,v_n\}, \{(v_i,v_j) : i \neq j, [a_i,b_i] \cap [a_j,b_j] \neq \emptyset\})$, où $[a_1,b_1],\dots,[a_n,b_n]$ est un ensemble d'intervalles fermés.
    (c) Les graphes triangulés (voir l'exercice 28 du chapitre 8).

∗ 16. Soit G un graphe non orienté. Prouver que les affirmations suivantes sont équivalentes :
    (a) G est parfait.
    (b) Pour toute fonction poids $c : V(G) \to \mathbb{Z}_+$ le poids maximum d'une clique dans G est égal au nombre minimum d'ensembles stables tel que chaque sommet v soit contenu dans $c(v)$ d'entre eux.
    (c) Pour toute fonction poids $c : V(G) \to \mathbb{Z}_+$ le poids maximum d'un ensemble stable dans G est égal au nombre minimum de cliques tel que chaque sommet v soit contenu dans $c(v)$ d'entre eux.
    (d) Le système d'inégalités définissant (16.3) est TDI.
    (e) Le polytope des cliques de G, c.-à-d. l'enveloppe convexe des vecteurs d'incidence de toutes les cliques de G, correspond à

$$\left\{ x \in \mathbb{R}_+^{V(G)} : \sum_{v \in S} x_v \leq 1 \text{ pour tous les ensembles stables } S \text{ de } G \right\}. \quad (16.9)$$

(f) Le système d'inégalités définissant (16.9) est TDI.
Remarque : le polytope (16.9) est appelé l'antibloquant du polytope (16.3).

17. Une instance de MAX-SAT est dite k-satisfaisable, si pour n'importe quel ensemble de k de ses clauses, elles peuvent être simultanément satisfaites. Soit r_k l'infimum de la fraction des clauses que l'on peut toujours satisfaire dans toute instance k-satisfaisable.
    (a) Prouver que $r_1 = \frac{1}{2}$.
    (*Indication :* théorème 16.28.)
    (b) Prouver que $r_2 = \frac{\sqrt{5}-1}{2}$.
    (*Indication :* certaines variables apparaissent dans des clauses formées d'un seul élément (sans perte de généralité, toutes les clauses constituées d'un seul élément sont positives), les mettre à *vrai* avec probabilité a (pour un certain $\frac{1}{2} < a < 1$), et mettre les autres variables à *vrai* avec probabilité $\frac{1}{2}$. Appliquer la technique de dérandomisation et choisir a de manière appropriée.)
    (c) Prouver que $r_3 \geq \frac{2}{3}$.
    (Lieberherr et Specker [1981])
18. Erdős [1967] a démontré le résultat suivant : pour chaque constante $k \in \mathbb{N}$, la meilleure fraction d'arêtes dont nous pouvons garantir qu'elles font partie d'une coupe maximum est égale à $\frac{1}{2}$, même si l'on se restreint aux graphes ne contenant pas de cycles impairs de longueur inférieure ou égale à k. (Comparer avec l'exercice 10(a).)
    (a) Que peut-on dire du cas où $k = \infty$?
    (b) Montrer comment le PROBLÈME DE LA COUPE MAXIMUM peut être réduit à MAX-SAT.
    Indication : utiliser une variable pour chaque sommet et deux clauses $\{x, y\}, \{\bar{x}, \bar{y}\}$ pour chaque arête (x, y).
    (c) Utiliser (b) et le théorème d'Erdős afin de prouver que $r_k \leq \frac{3}{4}$ pour tout k. (Pour une définition de r_k, voir exercice 17.)
    Remarque : Trevisan [2004] a prouvé que $\lim_{k \to \infty} r_k = \frac{3}{4}$.
19. Prouver que la probabilité d'erreur égale à $\frac{1}{2}$ dans la définition 16.31 peut être remplacée, de manière équivalente, par tout nombre compris entre 0 et 1. Déduire de cela (et de la preuve du théorème 16.34) qu'il n'existe pas d'algorithme d'approximation de facteur ρ pour le PROBLÈME DE LA CLIQUE MAXIMUM pour tout $\rho \geq 1$ (à moins que $P = NP$).
20. Prouver que le PROBLÈME DE LA CLIQUE MAXIMUM est L-réductible au PROBLÈME D'EMPILEMENT D'ENSEMBLES : étant donné un système d'ensembles $(U, \mathcal{S})$, trouver une sous-famille de cardinalité maximum $\mathcal{R} \subseteq \mathcal{S}$ dont les éléments soient deux à deux disjoints.

21. Prouver qu'il n'existe pas d'algorithme d'approximation absolue pour le PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS (à moins que $P = NP$).

22. Prouver que le problème MAX-2SAT est *MAXSNP*-difficile.
Indication : utiliser le corollaire 16.45.
(Papadimitriou et Yannakakis [1991])

Références

Littérature générale :

Asano, T., Iwama, K., Takada, H., Yamashita, Y. [2000] : Designing high-quality approximation algorithms for combinatorial optimization problems. IEICE Transactions on Communications/Electronics/Information and Systems E83-D (2000), 462–478

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M. [1999] : Complexity and Approximation : Combinatorial Optimization Problems and Their Approximability Properties. Springer, Berlin 1999

Garey, M.R., Johnson, D.S. [1979] : Computers and Intractability ; A Guide to the Theory of *NP*-Completeness. Freeman, San Francisco 1979, Chapter 4

Hochbaum, D.S. [1996] : Approximation Algorithms for *NP*-Hard Problems. PWS, Boston, 1996

Horowitz, E., Sahni, S. [1978] : Fundamentals of Computer Algorithms. Computer Science Press, Potomac 1978, Chapter 12

Shmoys, D.B. [1995] : Computing near-optimal solutions to combinatorial optimization problems. In : Combinatorial Optimization ; DIMACS Series in Discrete Mathematics and Theoretical Computer Science 20 (W. Cook, L. Lovász, P. Seymour, eds.), AMS, Providence 1995

Papadimitriou, C.H. [1994] : Computational Complexity, Addison-Wesley, Reading 1994, Chapter 13

Vazirani, V.V. [2001] : Approximation Algorithms. Springer, Berlin, 2001

Références citées :

Ajtai, M. [1994] : Recursive construction for 3-regular expanders. Combinatorica 14 (1994), 379–416

Alizadeh, F. [1995] : Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM Journal on Optimization 5 (1995), 13–51

Appel, K., Haken, W. [1977] : Every planar map is four colorable ; Part I ; Discharging. Illinois Journal of Mathematics 21 (1977), 429–490

Appel, K., Haken, W., Koch, J. [1977] : Every planar map is four colorable ; Part II ; Reducibility. Illinois Journal of Mathematics 21 (1977), 491–567

Arora, S. [1994] : Probabilistic checking of proofs and the hardness of approximation problems, Ph.D. thesis, U.C. Berkeley, 1994

Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M. [1998] : Proof verification and hardness of approximation problems. Journal of the ACM 45 (1998), 501–555

Arora, S., Safra, S. [1998] : Probabilistic checking of proofs. Journal of the ACM 45 (1998), 70–122

Asano, T. [2006] : An improved analysis of Goemans Williamson's LP-relaxation for MAX SAT. Theoretical Computer Science 354 (2006), 339–353

Bar-Yehuda, R., Even, S. [1981] : A linear-time approximation algorithm for the weighted vertex cover problem. Journal of Algorithms 2 (1981), 198–203

Becker, A., Geiger, D. [1996] : Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. Artificial Intelligence Journal 83 (1996), 1–22

Bellare, M., Sudan, M. [1994] : Improved non-approximability results. Proceedings of the 26th Annual ACM Symposium on the Theory of Computing (1994), 184–193

Bellare, M., Goldreich, O., Sudan, M. [1998] : Free bits, PCPs and nonapproximability – towards tight results. SIAM Journal on Computing 27 (1998), 804–915

Berge, C. [1961] : Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. Wissenschaftliche Zeitschrift, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe (1961), 114–115

Berge, C. [1962] : Sur une conjecture relative au problème des codes optimaux. Communication, 13ème assemblée générale de l'URSI, Tokyo 1962

Berman, P., Fujito, T. [1999] : On approximation properties of the independent set problem for low degree graphs. Theory of Computing Systems 32 (1999), 115–132

Brooks, R.L. [1941] : On colouring the nodes of a network. Proceedings of the Cambridge Philosophical Society 37 (1941), 194–197

Chen, J., Friesen, D.K., Zheng, H. [1999] : Tight bound on Johnson's algorithm for maximum satisfiability. Journal of Computer and System Sciences 58 (1999), 622–640

Chlebík, M. Chlebíková, J. [2006] : Complexity of approximating bounded variants of optimization problems. Theoretical Computer Science 354 (2006), 320-338

Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P., Vušković, K. [2005] : Recognizing Berge graphs. Combinatorica 25 (2005), 143–186

Chudnovsky, M., Robertson, N., Seymour, P., Thomas, R. [2006] : The strong perfect graph theorem. Annals of Mathematics 164 (2006), 51–229

Chvátal, V. [1975] : On certain polytopes associated with graphs. Journal of Combinatorial Theory B 18 (1975), 138–154

Chvátal, V. [1979] : A greedy heuristic for the set cover problem. Mathematics of Operations Research 4 (1979), 233–235

Clementi, A.E.F., Trevisan, L. [1999] : Improved non-approximability results for minimum vertex cover with density constraints. Theoretical Computer Science 225 (1999), 113–128

Deza, M.M., Laurent, M. [1997] : Geometry of Cuts and Metrics. Springer, Berlin 1997

Dinur, I. [2007] : The PCP theorem by gap amplification. Journal of the ACM 54 (2007), Article 12

Dinur, I., Safra, S. [2002] : On the hardness of approximating minimum vertex cover. Annals of Mathematics 162 (2005), 439–485

Erdős, P. [1967] : On bipartite subgraphs of graphs. Mat. Lapok. 18 (1967), 283–288

Feige, U. [1998] : A threshold of $\ln n$ for the approximating set cover. Journal of the ACM 45 (1998), 634–652

Feige, U. [2004] : Approximating maximum clique by removing subgraphs. SIAM Journal on Discrete Mathematics 18 (2004), 219–225

Feige, U., Goemans, M.X. [1995] : Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. Proceedings of the 3rd Israel Symposium on Theory of Computing and Systems (1995), 182–189

Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M. [1996] : Interactive proofs and the hardness of approximating cliques. Journal of the ACM 43 (1996), 268–292

Fernández-Baca, D., Lagergren, J. [1998] : On the approximability of the Steiner tree problem in phylogeny. Discrete Applied Mathematics 88 (1998), 129–145

Fulkerson, D.R. [1972] : Anti-blocking polyhedra. Journal of Combinatorial Theory B 12 (1972), 50–71

Fürer, M., Raghavachari, B. [1994] : Approximating the minimum-degree Steiner tree to within one of optimal. Journal of Algorithms 17 (1994), 409–423

Garey, M.R., Johnson, D.S. [1976] : The complexity of near-optimal graph coloring. Journal of the ACM 23 (1976), 43–49

Garey, M.R., Johnson, D.S., Stockmeyer, L. [1976] : Some simplified *NP*-complete graph problems. Theoretical Computer Science 1 (1976), 237–267

Goemans, M.X., Williamson, D.P. [1994] : New 3/4-approximation algorithms for the maximum satisfiability problem. SIAM Journal on Discrete Mathematics 7 (1994), 656–666

Goemans, M.X., Williamson, D.P. [1995] : Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42 (1995), 1115–1145

Grötschel, M., Lovász, L., Schrijver, A. [1988] : Geometric Algorithms and Combinatorial Optimization. Springer, Berlin 1988

Halldórsson, M.M., Radhakrishnan, J. [1997] : Greed is good : approximating independent sets in sparse and bounded degree graphs. Algorithmica 18 (1997), 145–163

Håstad, J. [2001] : Some optimal inapproximability results. Journal of the ACM 48 (2001), 798–859

Heawood, P.J. [1890] : Map colour theorem. Quarterly Journal of Pure Mathematics 24 (1890), 332–338

Hochbaum, D.S. [1982] : Approximation algorithms for the set covering and vertex cover problems. SIAM Journal on Computing 11 (1982), 555–556

Hochbaum, D.S., Shmoys, D.B. [1985] : A best possible heuristic for the k-center problem. Mathematics of Operations Research 10 (1985), 180–184

Holyer, I. [1981] : The *NP*-completeness of edge-coloring. SIAM Journal on Computing 10 (1981), 718–720

Hougardy, S., Prömel, H.J., Steger, A. [1994] : Probabilistically checkable proofs and their consequences for approximation algorithms. Discrete Mathematics 136 (1994), 175–223

Hsu, W.L., Nemhauser, G.L. [1979] : Easy and hard bottleneck location problems. Discrete Applied Mathematics 1 (1979), 209–216

Johnson, D.S. [1974] : Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences 9 (1974), 256–278

Khanna, S., Linial, N., Safra, S. [2000] : On the hardness of approximating the chromatic number. Combinatorica 20 (2000), 393–415

Knuth, D.E. [1969] : The Art of Computer Programming ; Vol. 2. Addison-Wesley, Reading 1969 (3rd edition : 1997)

König, D. [1916] : Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. Mathematische Annalen 77 (1916), 453–465

Lieberherr, K., Specker, E. [1981] : Complexity of partial satisfaction. Journal of the ACM 28 (1981), 411–421

Lovász, L. [1972] : Normal hypergraphs and the perfect graph conjecture. Discrete Mathematics 2 (1972), 253–267

Lovász, L. [1975] : On the ratio of optimal integral and fractional covers. Discrete Mathematics 13 (1975), 383–390

Lovász, L. [1979a] : On the Shannon capacity of a graph. IEEE Transactions on Information Theory 25 (1979), 1–7

Lovász, L. [1979b] : Graph theory and integer programming. In : Discrete Optimization I ; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, eds.), North-Holland, Amsterdam 1979, pp. 141–158

Lovász, L. [2003] : Semidefinite programs and combinatorial optimization. In : Recent Advances in Algorithms and Combinatorics (B.A. Reed, C.L. Sales, eds.), Springer, New York (2003), pp. 137–194

Mahajan, S., Ramesh, H. [1999] : Derandomizing approximation algorithms based on semidefinite programming. SIAM Journal on Computing 28 (1999), 1641–1663

Papadimitriou, C.H., Steiglitz, K. [1982] : Combinatorial Optimization ; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, pp. 406–408

Papadimitriou, C.H., Yannakakis, M. [1991] : Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43 (1991), 425–440

Papadimitriou, C.H., Yannakakis, M. [1993] : The traveling salesman problem with distances one and two. Mathematics of Operations Research 18 (1993), 1–12

Raghavan, P., Thompson, C.D. [1987] : Randomized rounding : a technique for provably good algorithms and algorithmic proofs. Combinatorica 7 (1987), 365–374

Raz, R., Safra, S. [1997] : A sub constant error probability low degree test, and a sub constant error probability *PCP* characterization of *NP*. Proceedings of the 29th Annual ACM Symposium on Theory of Computing (1997), 475–484

Robertson, N., Sanders, D.P., Seymour, P., Thomas, R. [1997] : The four colour theorem. Journal of Combinatorial Theory B 70 (1997), 2–44

Robertson, N., Sanders, D.P., Seymour, P., Thomas, R. [1996] : Efficiently four-coloring planar graphs. Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (1996), 571–575

Sanders, P., Steurer, D. [2005] : An asymptotic approximation scheme for multigraph edge coloring. Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (2005), 897–906

Singh, M. Lau, L.C. [2007] : Approximating minimum bounded degree spanning trees to within one of optimal. Proceedings of the 39th Annual ACM Symposium on Theory of Computing (2007), 661–670

Slavík, P. [1997] : A tight analysis of the greedy algorithm for set cover. Journal of Algorithms 25 (1997), 237–254

Stockmeyer, L.J. [1973] : Planar 3-colorability is polynomial complete. ACM SIGACT News 5 (1973), 19–25

Trevisan, L. [2004] : On local versus global satisfiability. SIAM Journal on Discrete Mathematics 17 (2004), 541–547

Vizing, V.G. [1964] : On an estimate of the chromatic class of a p-graph. Diskret. Analiz 3 (1964), 23–30 [in Russian]

Wigderson, A. [1983] : Improving the performance guarantee for approximate graph coloring. Journal of the ACM 30 (1983), 729–735

Yannakakis, M. [1994] : On the approximation of maximum satisfiability. Journal of Algorithms 17 (1994), 475–502

Zuckerman, D. [2006] : Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. Proceedings of the 38th Annual ACM Symposium on Theory of Computing (2006), 681–690

Chapitre 17

Le problème du sac à dos

Le PROBLÈME DU COUPLAGE PARFAIT DE POIDS MINIMUM et le PROBLÈME D'INTERSECTION DE MATROÏDES AVEC POIDS présentés aux chapitres précédents font partie des problèmes les plus «difficiles» pour lesquels on connaît un algorithme polynomial. Dans ce chapitre, nous traitons le problème suivant qui se révèle être, en un certain sens, le plus «facile» des problèmes *NP*-difficiles :

PROBLÈME DU SAC À DOS	
Instance	Des entiers non négatifs n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ et W.
Tâche	Trouver un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} w_j \leq W$ et tel que $\sum_{j \in S} c_j$ soit maximum.

Les applications correspondent aux situations où l'on veut sélectionner un sous-ensemble optimal, de poids total borné, d'un ensemble d'éléments ayant chacun un poids et un profit associés.

Nous commençons par étudier la version fractionnaire de ce problème au paragraphe 17.1, qui se révéle être résoluble en temps linéaire. Le PROBLÈME DU SAC À DOS en nombres entiers est *NP*-difficile comme nous le montrerons au paragraphe 17.2, mais un algorithme pseudo-polynomial le résout optimalement. Nous montrerons au paragraphe 17.3 que cet algorithme combiné à une technique d'arrondi peut être utilisé pour concevoir un schéma d'approximation entièrement polynomial.

17.1 Sac à dos fractionnaire et problème du médian pondéré

Nous considérons le problème suivant :

PROBLÈME DU SAC À DOS FRACTIONNAIRE	
Instance	Des entiers non négatifs n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ et W.
Tâche	Trouver des nombres $x_1, \ldots, x_n \in [0,1]$ tels que $\sum_{j=1}^{n} x_j w_j \leq W$ et tel que $\sum_{j=1}^{n} x_j c_j$ soit maximum.

L'observation suivante suggère un algorithme simple qui nécessite de trier les éléments de manière appropriée :

Proposition 17.1. (Dantzig [1957]) *Soient $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ et W des entiers non négatifs tels que $\sum_{i=1}^{n} w_i > W$ et*

$$\frac{c_1}{w_1} \;\geq\; \frac{c_2}{w_2} \;\geq\; \cdots \;\geq\; \frac{c_n}{w_n},$$

et considérons

$$k \;:=\; \min\left\{ j \in \{1,\ldots,n\} : \sum_{i=1}^{j} w_i > W \right\}.$$

Alors une solution optimale de l'instance considérée du PROBLÈME DU SAC À DOS FRACTIONNAIRE *est définie par*

$$\begin{aligned} x_j &:= 1 && \text{pour } j = 1, \ldots, k-1, \\ x_k &:= \frac{W - \sum_{j=1}^{k-1} w_j}{w_k}, \\ x_j &:= 0 && \text{pour } j = k+1, \ldots, n. \end{aligned}$$

□

Trier les éléments requiert un temps $O(n \log n)$ (théorème 1.5) et calculer k peut être fait en temps $O(n)$ par un simple balayage linéaire. Bien que cet algorithme soit assez rapide, on peut faire encore mieux. Observons que le problème se réduit à une recherche de médian pondéré :

Définition 17.2. *Soient $n \in \mathbb{N}$, $z_1, \ldots, z_n \in \mathbb{R}$, $w_1, \ldots, w_n \in \mathbb{R}_+$ et $W \in \mathbb{R}$ tel que $0 < W \leq \sum_{i=1}^{n} w_i$. Alors le $(w_1, \ldots, w_n; W)$*-**médian pondéré par rapport à** $(z_1, \ldots, z_n)$ *est défini comme étant l'unique nombre z^* pour lequel*

$$\sum_{i:z_i<z^*} w_i \;<\; W \;\leq\; \sum_{i:z_i\leq z^*} w_i.$$

Nous devons donc résoudre le problème suivant :

PROBLÈME DU MÉDIAN PONDÉRÉ	
Instance	Un entier n, des nombres $z_1, \ldots, z_n \in \mathbb{R}$, $w_1, \ldots, w_n \in \mathbb{R}_+$ et un nombre W tel que $0 < W \leq \sum_{i=1}^{n} w_i$.
Tâche	Trouver le $(w_1, \ldots, w_n; W)$-médian pondéré par rapport à $(z_1, \ldots, z_n)$.

Le problème suivant est un cas particulier important :

PROBLÈME DE SÉLECTION	
Instance	Un entier n, des nombres $z_1, \ldots, z_n \in \mathbb{R}$, et un entier $k \in \{1, \ldots, n\}$.
Tâche	Trouver les k plus petits nombres parmi $z_1, \ldots, z_n$.

Le médian pondéré peut être déterminé en temps $O(n)$: l'algorithme suivant est une version pondérée de l'algorithme de Blum *et al.* [1973] ; voir aussi Vygen [1997].

ALGORITHME DU MÉDIAN PONDÉRÉ	
Input	Un entier n, des nombres $z_1, \ldots, z_n \in \mathbb{R}$, $w_1, \ldots, w_n \in \mathbb{R}_+$ et un nombre W tel que $0 < W \leq \sum_{i=1}^{n} w_i$.
Output	Le $(w_1, \ldots, w_n; W)$-médian pondéré par rapport à $(z_1, \ldots, z_n)$.

① Partitionner la liste $z_1, \ldots, z_n$ en blocs composés chacun de cinq éléments. Trouver le médian (non pondéré) de chaque bloc. Soit M la liste de ces $\left\lceil \frac{n}{5} \right\rceil$ éléments médians.

② Trouver (récursivement) le médian non pondéré de M, le noter z_m.

③ Comparer chaque élément à z_m. Sans perte de généralité, $z_i < z_m$ pour $i = 1, \ldots, k$, $z_i = z_m$ pour $i = k+1, \ldots, l$ et $z_i > z_m$ pour $i = l+1, \ldots, n$.

④ **If** $\sum_{i=1}^{k} w_i < W \leq \sum_{i=1}^{l} w_i$ **then stop** ($z^* := z_m$).

If $\sum_{i=1}^{l} w_i < W$ **then** trouver récursivement le $\left(w_{l+1}, \ldots, w_n;\ W - \sum_{i=1}^{l} w_i\right)$- médian pondéré par rapport à $(z_{l+1}, \ldots, z_n)$. **Stop**.

If $\sum_{i=1}^{k} w_i \geq W$ **then** trouver récursivement le $(w_1, \ldots, w_k; W)$-médian pondéré par rapport à $(z_1, \ldots, z_k)$. **Stop**.

Théorème 17.3. *L'*ALGORITHME DU MÉDIAN PONDÉRÉ *répond correctement et nécessite seulement un temps* $O(n)$.

Preuve. L'exactitude de l'algorithme est facilement vérifiée. Pour n éléments, notons $f(n)$ le temps de calcul dans le pire des cas. On obtient

$$f(n) \;=\; O(n) + f\left(\left\lceil \frac{n}{5} \right\rceil\right) + O(n) + f\left(\frac{1}{2}\left\lceil \frac{n}{5} \right\rceil 5 + \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil 2\right),$$

car l'appel récursif à l'étape ④ omet au moins trois éléments d'au moins la moitié des blocs constitués de cinq éléments. La formule récursive précédente implique que $f(n) = O(n)$: comme $\lceil \frac{n}{5} \rceil \leq \frac{9}{41}n$ pour tout $n \geq 37$, on obtient $f(n) \leq cn + f\left(\frac{9}{41}n\right) + f\left(\frac{7}{2}\frac{9}{41}n\right)$ pour un c convenable et $n \geq 37$. On peut alors vérifier facilement par induction que $f(n) \leq (82c + f(36))n$. Le temps total de calcul est donc bien linéaire. □

Nous obtenons de manière immédiate les corollaires suivants :

Corollaire 17.4. (Blum *et al.* [1973]) *Le* PROBLÈME DE SÉLECTION *peut être résolu en temps* $O(n)$.

Preuve. On pose $w_i := 1$ pour $i = 1, \ldots, n$ et $W := k$ et on applique le théorème 17.3. □

Corollaire 17.5. *Le* PROBLÈME DU SAC À DOS FRACTIONNAIRE *peut être résolu en temps linéaire.*

Preuve. Comme nous l'avons remarqué au début de cette section, en posant $z_i := -\frac{c_i}{w_i}$ $(i = 1, \ldots, n)$, nous réduisons le PROBLÈME DU SAC À DOS FRACTIONNAIRE au PROBLÈME DU MÉDIAN PONDÉRÉ. □

17.2 Un algorithme pseudo-polynomial

Nous passons maintenant au PROBLÈME DU SAC À DOS (en nombres entiers). Les techniques du paragraphe précédent sont également d'une certaine utilité ici :

Proposition 17.6. *Soient* $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ *et* W *des entiers non négatifs tels que* $w_j \leq W$ *pour* $j = 1, \ldots, n$, $\sum_{i=1}^{n} w_i > W$, *et*

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \cdots \geq \frac{c_n}{w_n}.$$

Soit

$$k := \min\left\{ j \in \{1, \ldots, n\} : \sum_{i=1}^{j} w_i > W \right\}.$$

Alors l'algorithme consistant à choisir la meilleure des deux solutions réalisables $\{1, \ldots, k-1\}$ *et* $\{k\}$ *constitue un algorithme d'approximation de facteur 2 pour le* PROBLÈME DU SAC À DOS *de temps de calcul* $O(n)$.

Preuve. Étant donné une instance du PROBLÈME DU SAC À DOS, les éléments $i \in \{1, \ldots, n\}$ tels que $w_i > W$ sont inutiles et peuvent être supprimés préalablement. Si maintenant $\sum_{i=1}^{n} w_i \leq W$, alors $\{1, \ldots, n\}$ est une solution optimale. Sinon on calcule le nombre k en temps $O(n)$ sans trier : cela est juste un PROBLÈME DE MÉDIAN PONDÉRÉ comme ci-dessus (théorème 17.3).

D'après la proposition 17.1, $\sum_{i=1}^{k} c_i$ est une borne supérieure de la valeur optimale du PROBLÈME DU SAC À DOS FRACTIONNAIRE, et donc également de la valeur optimale du PROBLÈME DU SAC À DOS en nombres entiers. On voit donc que la meilleure des deux solutions réalisables $\{1, \ldots, k-1\}$ et $\{k\}$ réalise au moins la moitié de la valeur optimale. □

Mais on s'intéresse davantage à une solution exacte du PROBLÈME DU SAC À DOS. Cependant, nous devons faire l'observation suivante :

Théorème 17.7. *Le* PROBLÈME DU SAC À DOS *est NP-difficile.*

Preuve. Nous prouvons que le problème de décision suivant est *NP*-complet : étant donné des entiers non négatifs n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$, W et K, existe-t-il un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} w_j \leq W$ et $\sum_{j \in S} c_j \geq K$?

Ce problème de décision appartient de manière évidente à *NP*. Pour montrer qu'il est *NP*-complet, nous montrons qu'il s'obtient par transformation du problème de la SOMME DE SOUS-ENSEMBLES (voir corollaire 15.27). Étant donné une instance $c_1, \ldots, c_n, K$ du problème de la SOMME DE SOUS-ENSEMBLES, définissons $w_j := c_j \ \ (j = 1, \ldots, n)$ et $W := K$. Cela fournit de manière évidente une instance équivalente du problème de décision précédent. □

Comme nous n'avons pas démontré que le PROBLÈME DU SAC À DOS est fortement *NP*-difficile, on peut espérer qu'il existe un algorithme pseudo-polynomial. En effet, l'algorithme donné dans la preuve du théorème 15.38 peut être facilement généralisé en introduisant des poids sur les arêtes et en résolvant un problème de plus court chemin. Cela fournit un algorithme de complexité $O(nW)$ (exercice 3).

De manière similaire, on peut également obtenir un algorithme de temps de calcul $O(nC)$, où $C := \sum_{j=1}^{n} c_j$. Nous décrivons cet algorithme de manière directe, c.-à-d. sans construire un graphe et des plus courts chemins. L'exactitude de l'algorithme reposant sur de simples formules récursives, nous dirons qu'il s'agit d'un algorithme de programmation dynamique. Il est essentiellement dû à Bellman [1956,1957] et Dantzig [1957].

ALGORITHME DE PROGRAMMATION DYNAMIQUE POUR LE PROBLÈME DU SAC À DOS

Input Des entiers non négatifs n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ et W.

Output Un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} w_j \leq W$ et que $\sum_{j \in S} c_j$ soit maximum.

① Soit C une borne supérieure de la valeur de la solution optimale, par exemple,
$C := \sum_{j=1}^{n} c_j$.

② Poser $x(0,0) := 0$ et $x(0,k) := \infty$ pour $k = 1, \ldots, C$.

③ **For** $j := 1$ **to** n **do** :
  For $k := 0$ **to** C **do** :
    Poser $s(j,k) := 0$ et $x(j,k) := x(j-1,k)$.
  For $k := c_j$ **to** C **do** :
    If $x(j-1,k-c_j) + w_j \leq \min\{W, x(j,k)\}$ **then** :
      Poser $x(j,k) := x(j-1,k-c_j) + w_j$ et $s(j,k) := 1$.

④ Poser $k = \max\{i \in \{0,\dots,C\} : x(n,i) < \infty\}$. Poser $S := \emptyset$.
For $j := n$ **down to** 1 **do** :
  If $s(j,k) = 1$ **then** poser $S := S \cup \{j\}$ et $k := k - c_j$.

Théorème 17.8. *L'*ALGORITHME DE PROGRAMMATION DYNAMIQUE POUR LE PROBLÈME DU SAC À DOS *trouve une solution optimale en temps* $O(nC)$.

Preuve. Le temps de calcul est évident.

La variable $x(j,k)$ désigne le poids total minimum d'un sous-ensemble $S \subseteq \{1,\dots,j\}$ tel que $\sum_{i\in S} w_i \leq W$ et $\sum_{i\in S} c_i = k$. L'algorithme calcule correctement ces valeurs en utilisant les formules récursives

$$x(j,k) = \begin{cases} x(j-1,k-c_j) + w_j & \text{si } c_j \leq k \text{ et} \\ & x(j-1,k-c_j) + w_j \leq \min\{W, x(j-1,k)\} \\ x(j-1,k) & \text{sinon} \end{cases}$$

pour $j = 1,\dots,n$ et $k = 0,\dots,C$. Les variables $s(j,k)$ indiquent lequel de ces deux cas s'applique. Ainsi l'algorithme énumère tous les sous-ensembles $S \subseteq \{1,\dots,n\}$ sauf ceux qui sont réalisables ou ceux qui sont dominés par d'autres : on dit que S est dominé par S' si $\sum_{j\in S} c_j = \sum_{j\in S'} c_j$ et $\sum_{j\in S} w_j \geq \sum_{j\in S'} w_j$. À l'étape ④, on choisit le meilleur sous-ensemble réalisable. □

Bien entendu, on préférerait avoir une meilleure borne supérieure C que $\sum_{i=1}^{n} c_i$. Par exemple, l'algorithme d'approximation de facteur 2 de la proposition 17.6 peut être utilisé ; en multipliant par 2 la valeur fournie par l'algorithme, on obtient une borne supérieure de la valeur optimale. Nous utiliserons cette idée plus tard.

La borne en $O(nC)$ n'est pas polynomiale par rapport à la taille de l'input, car la taille de l'input peut seulement être bornée par $O(n \log\ C + n \log\ W)$ (on suppose que $w_j \leq W$ pour tout j). Mais on a un algorithme pseudo-polynomial qui peut être assez efficace si les nombres impliqués ne sont pas trop grands. Si les poids $w_1,\dots,w_n$ et les profits $c_1,\dots,c_n$ sont petits, l'algorithme en $O(nc_{\max}w_{\max})$ de Pisinger [1999] est le plus rapide (avec $c_{\max} := \max\{c_1,\dots,c_n\}$ et $w_{\max} := \max\{w_1,\dots,w_n\}$).

17.3 Un schéma d'approximation entièrement polynomial

Dans ce paragraphe, on s'intéresse à l'existence d'algorithmes d'approximation pour le PROBLÈME DU SAC À DOS. D'après la proposition 16.23, le PROBLÈME DU SAC À DOS n'a pas d'algorithme d'approximation absolue à moins que *P = NP*.

Cependant, nous prouverons que le PROBLÈME DU SAC À DOS a un schéma d'approximation entièrement polynomial. Un tel algorithme a été trouvé en premier par Ibarra et Kim [1975].

Puisque le temps de calcul de l'ALGORITHME DE PROGRAMMATION DYNAMIQUE POUR LE PROBLÈME DU SAC À DOS dépend de C, une idée naturelle est de diviser tous les nombres $c_1, \ldots, c_n$ par 2 et de les arrondir vers le bas. Ceci réduira le temps de calcul, mais peut amener à des solutions imprécises. De manière plus générale, en posant

$$\bar{c_j} := \left\lfloor \frac{c_j}{t} \right\rfloor \qquad (j = 1, \ldots, n)$$

on réduit le temps de calcul par un facteur t. «Échanger» de la précision contre du temps de calcul est typique des schémas d'approximation. Pour $S \subseteq \{1, \ldots, n\}$ on écrit $c(S) := \sum_{i \in S} c_i$.

SCHÉMA D'APPROXIMATION POUR LE PROBLÈME DU SAC À DOS

Input Des entiers non négatifs $n, c_1, \ldots, c_n, w_1, \ldots, w_n$ et W. Un nombre $\epsilon > 0$.

Output Un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} w_j \leq W$ et $\sum_{j \in S} c_j \geq \frac{1}{1+\epsilon} \sum_{j \in S'} c_j$ pour tout $S' \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S'} w_j \leq W$.

① Exécuter l'algorithme d'approximation de facteur 2 de la proposition 17.6. Soit S_1 la solution obtenue. **If** $c(S_1) = 0$ **then** poser $S := S_1$ et **stop**.

② Poser $t := \max\left\{1, \frac{\epsilon c(S_1)}{n}\right\}$.
Poser $\bar{c_j} := \left\lfloor \frac{c_j}{t} \right\rfloor$ pour $j = 1, \ldots, n$.

③ Appliquer l'ALGORITHME DE PROGRAMMATION DYNAMIQUE POUR LE PROBLÈME DU SAC À DOS à l'instance $(n, \bar{c}_1, \ldots, \bar{c}_n, w_1, \ldots, w_n, W)$; poser $C := \frac{2c(S_1)}{t}$. Soit S_2 la solution obtenue.

④ **If** $c(S_1) > c(S_2)$ **then** poser $S := S_1$, **else** poser $S := S_2$.

Théorème 17.9. (Ibarra et Kim [1975], Sahni [1976], Gens et Levner [1979]) *Le* SCHÉMA D'APPROXIMATION POUR LE PROBLÈME DU SAC À DOS *est un schéma d'approximation entièrement polynomial ; son temps de calcul est* $O\left(n^2 \cdot \frac{1}{\epsilon}\right)$.

Preuve. Si l'algorithme s'arrête à l'étape ①, alors, d'après la proposition 17.6, S_1 est optimal. Supposons donc maintenant $c(S_1) > 0$. Soit S^* une solution optimale de l'instance de départ. Puisque, d'après la proposition 17.6, $2c(S_1) \geq c(S^*)$, C à l'étape ③ est une borne supérieure convenable de la valeur de la solution optimale de l'instance modifiée. Alors S_2 est une solution optimale de l'instance modifiée, d'après le théorème 17.8. On a ainsi :

$$\sum_{j \in S_2} c_j \geq \sum_{j \in S_2} t\bar{c_j} = t \sum_{j \in S_2} \bar{c_j} \geq t \sum_{j \in S^*} \bar{c_j} = \sum_{j \in S^*} t\bar{c_j} > \sum_{j \in S^*} (c_j - t) \geq c(S^*) - nt.$$

Si $t = 1$, alors S_2 est optimal d'après le théorème 17.8. Sinon l'inégalité précédente implique $c(S_2) \geq c(S^*) - \epsilon c(S_1)$, et on peut conclure que

$$(1+\epsilon)c(S) \;\geq\; c(S_2) + \epsilon c(S_1) \;\geq\; c(S^*).$$

On a donc un algorithme d'approximation de facteur $(1+\epsilon)$ pour tout $\epsilon > 0$ fixé. D'après le théorème 17.8, le temps de calcul de l'étape ③ peut être borné par

$$O(nC) \;=\; O\left(\frac{nc(S_1)}{t}\right) \;=\; O\left(n^2 \cdot \frac{1}{\epsilon}\right).$$

Les autres étapes peuvent facilement être exécutées en temps $O(n)$. □

Lawler [1979] a trouvé un schéma d'approximation entièrement polynomial similaire dont le temps de calcul est en $O\left(n \log\left(\frac{1}{\epsilon}\right) + \frac{1}{\epsilon^4}\right)$. Cela a été amélioré par Kellerer et Pferschy [2004].

Malheureusement, il n'existe pas beaucoup de problèmes ayant un schéma d'approximation entièrement polynomial. Pour rendre cela plus précis, considérons le PROBLÈME DE MAXIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE.

Nous avons utilisé une certaine relation de dominance dans notre construction de l'ALGORITHME DE PROGRAMMATION DYNAMIQUE et du SCHÉMA D'APPROXIMATION POUR LE PROBLÈME DU SAC À DOS. Nous généralisons ce concept de la façon suivante :

Définition 17.10. *Étant donné un système d'indépendance $(E, \mathcal{F})$, une fonction coût $c : E \to \mathbb{Z}_+$, des sous-ensembles $S_1, S_2 \subseteq E$, et $\epsilon > 0$. S_1* **ϵ-domine** *S_2 si*

$$\frac{1}{1+\epsilon}\, c(S_1) \;\leq\; c(S_2) \;\leq\; (1+\epsilon)\, c(S_1)$$

et s'il existe une base B_1 telle que $S_1 \subseteq B_1$ et que pour chaque base B_2 telle que $S_2 \subseteq B_2$ on ait

$$(1+\epsilon)\, c(B_1) \;\geq\; c(B_2).$$

PROBLÈME DE ϵ-DOMINANCE	
Instance	Un système d'indépendance $(E, \mathcal{F})$, une fonction coût $c : E \to \mathbb{Z}_+$, un nombre $\epsilon > 0$, et deux sous-ensembles $S_1, S_2 \subseteq E$.
Question	S_1 ϵ-domine-t-il S_2 ?

Bien entendu, le système d'indépendance est donné par un oracle, un oracle d'indépendance par exemple. L'ALGORITHME DE PROGRAMMATION DYNAMIQUE POUR LE PROBLÈME DU SAC À DOS fait un usage fréquent de 0-dominance. Il apparaît que l'existence d'un algorithme efficace pour le PROBLÈME DE ϵ-DOMINANCE est essentielle à l'existence d'un schéma d'approximation entièrement polynomial.

Théorème 17.11. (Korte et Schrader [1981]) *Soit $\mathcal{I}$ une famille de systèmes d'indépendance. Soit $\mathcal{I}'$ la famille des instances $(E, \mathcal{F}, c)$ du* PROBLÈME DE MAXIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE *telle que $(E, \mathcal{F}) \in \mathcal{I}$ et $c : E \to \mathbb{Z}_+$, et soit $\mathcal{I}''$ la famille des instances $(E, \mathcal{F}, c, \epsilon, S_1, S_2)$ du* PROBLÈME DE ϵ-DOMINANCE *telle que $(E, \mathcal{F}) \in \mathcal{I}$.*

Alors il existe un schéma d'approximation entièrement polynomial pour le PROBLÈME DE MAXIMISATION POUR DES SYSTÈMES D'INDÉPENDANCE *restreint à $\mathcal{I}'$ si et seulement s'il existe un algorithme pour le* PROBLÈME DE ϵ-DOMINANCE *restreint à $\mathcal{I}''$ dont le temps de calcul soit borné par un polynôme par rapport à la longueur de l'input et à $\frac{1}{\epsilon}$.*

Alors que la condition suffisante se prouve en généralisant le SCHÉMA D'APPROXIMATION POUR LE PROBLÈME DU SAC À DOS (exercice 10), la preuve de la condition nécessaire est assez compliquée et n'est pas présentée ici. La conclusion est que s'il existe un schéma d'approximation entièrement polynomial, alors une simple modification du SCHÉMA D'APPROXIMATION POUR LE PROBLÈME DU SAC À DOS convient. Voir aussi Woeginger [2000] pour un résultat similaire.

Afin de prouver que, pour un certain problème d'optimisation, il n'existe pas de schéma d'approximation entièrement polynomial, le théorème suivant est souvent plus utile :

Théorème 17.12. (Garey et Johnson [1978]) *Un problème d'optimisation fortement NP-difficile avec une fonction objectif entière, qui vérifie l'inégalité*

$$\mathrm{OPT}(I) \;\leq\; p\left(\mathrm{taille}(I), \mathrm{large}(I)\right)$$

pour un certain polynôme p et pour toutes les instances I, a un schéma d'approximation entièrement polynomial seulement si $P = NP$.

Preuve. Supposons que ce problème ait un schéma d'approximation entièrement polynomial. Appliquons ce schéma avec

$$\epsilon \;=\; \frac{1}{p(\mathrm{taille}(I), \mathrm{large}(I)) + 1}$$

et nous obtenons un algorithme pseudo-polynomial exact. D'après la proposition 15.40, cela est impossible à moins que $P = NP$. □

Exercices

1. Considérons le PROBLÈME DU SAC À DOS multiple fractionnaire défini de la manière suivante. Une instance est constituée d'entiers non négatifs m et n, de nombres w_j, c_{ij} et W_i ($1 \leq i \leq m$, $1 \leq j \leq n$). L'objectif est alors de trouver des nombres $x_{ij} \in [0, 1]$ tels que $\sum_{i=1}^{m} x_{ij} = 1$ pour tout j et $\sum_{j=1}^{n} x_{ij} w_j \leq W_i$ pour tout i et tels que $\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} c_{ij}$ soit minimum.
   Peut-on trouver un algorithme combinatoire polynomial pour ce problème (sans utiliser la PROGRAMMATION LINÉAIRE) ?
   Indication : réduire ce problème au PROBLÈME DU FLOT DE COÛT MINIMUM.

2. Considérons l'algorithme glouton suivant pour le PROBLÈME DU SAC À DOS (similaire à celui de la proposition 17.6). Trier les indices de telle façon que $\frac{c_1}{w_1} \geq \cdots \geq \frac{c_n}{w_n}$. Poser $S := \emptyset$. Pour $i := 1$ à n faire : si $\sum_{j \in S \cup \{i\}} w_j \leq W$ alors poser $S := S \cup \{i\}$. Montrer que ce n'est pas un algorithme d'approximation de facteur k quelle que soit la valeur de k.
3. Trouver un algorithme exact en $O(nW)$ pour le PROBLÈME DU SAC À DOS.
4. Considérons le problème suivant : étant donné des entiers non négatifs n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$ et W, trouver un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} w_j \geq W$ et que $\sum_{j \in S} c_j$ soit minimum. Comment peut-on résoudre ce problème à l'aide d'un algorithme pseudo-polynomial ?

∗ 5. Peut-on résoudre le PROBLÈME DU SAC À DOS multiple en nombres entiers (voir exercice 1) en temps pseudo-polynomial si m est fixé ?

6. Soient $c \in \{0, \ldots, k\}^m$ et $s \in [0, 1]^m$. Comment peut-on décider en temps $O(mk)$ si $\max\{cx : x \in \mathbb{Z}_+^m,\ sx \leq 1\} \leq k$?
7. Considérons les deux relaxations lagrangiennes de l'exercice 21 du chapitre 5. Montrer que l'une d'entre elles peut être résolue en temps linéaire alors que l'autre se réduit à m instances du PROBLÈME DU SAC À DOS.
8. Soit $m \in \mathbb{N}$ une constante. Considérons le problème d'emploi du temps suivant : étant donné n tâches et m machines, des coûts $c_{ij} \in \mathbb{Z}_+$ ($i = 1, \ldots, n$, $j = 1, \ldots, m$), et des capacités $T_j \in \mathbb{Z}_+$ ($j = 1, \ldots, m$), trouver une affectation $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$ telle que $|\{i \in \{1, \ldots, n\} : f(i) = j\}| \leq T_j$ pour $j = 1, \ldots, m$, et que le coût total $\sum_{i=1}^{n} c_{if(i)}$ soit minimum.
   Montrer que ce problème a un schéma d'approximation entièrement polynomial.
9. Donner un algorithme polynomial pour le PROBLÈME DE ϵ-DOMINANCE restreint aux matroïdes.

∗ 10. Prouver la condition nécessaire du théorème 17.11.

Références

Littérature générale :

Garey, M.R., Johnson, D.S. [1979] : Computers and Intractability ; A Guide to the Theory of *NP*-Completeness. Freeman, San Francisco 1979, Chapter 4

Martello, S., Toth, P. [1990] : Knapsack Problems ; Algorithms and Computer Implementations. Wiley, Chichester 1990

Papadimitriou, C.H., Steiglitz, K. [1982] : Combinatorial Optimization ; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Sections 16.2, 17.3, and 17.4

Références citées :

Bellman, R. [1956] : Notes on the theory of dynamic programming IV – maximization over discrete sets. Naval Research Logistics Quarterly 3 (1956), 67–70

Bellman, R. [1957] : Comment on Dantzig's paper on discrete variable extremum problems. Operations Research 5 (1957), 723–724

Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., Tarjan, R.E. [1973] : Time bounds for selection. Journal of Computer and System Sciences 7 (1973), 448–461

Dantzig, G.B. [1957] : Discrete variable extremum problems. Operations Research 5 (1957), 266–277

Garey, M.R., Johnson, D.S. [1978] : Strong *NP*-completeness results : motivation, examples, and implications. Journal of the ACM 25 (1978), 499–508

Gens, G.V., Levner, E.V. [1979] : Computational complexity of approximation algorithms for combinatorial problems. In : Mathematical Foundations of Computer Science ; LNCS 74 (J. Becvar, ed.), Springer, Berlin 1979, pp. 292–300

Ibarra, O.H., Kim, C.E. [1975] : Fast approximation algorithms for the knapsack and sum of subset problem. Journal of the ACM 22 (1975), 463–468

Kellerer, H., Pferschy, U. [2004] : Improved dynamic programming in connection with an FPTAS for the knapsack problem. Journal on Combinatorial Optimization 8 (2004), 5–11

Korte, B., Schrader, R. [1981] : On the existence of fast approximation schemes. In : Nonlinear Programming ; Vol. 4 (O. Mangaserian, R.R. Meyer, S.M. Robinson, eds.), Academic Press, New York 1981, pp. 415–437

Lawler, E.L. [1979] : Fast approximation algorithms for knapsack problems. Mathematics of Operations Research 4 (1979), 339–356

Pisinger, D. [1999] : Linear time algorithms for knapsack problems with bounded weights. Journal of Algorithms 33 (1999), 1–14

Sahni, S. [1976] : Algorithms for scheduling independent tasks. Journal of the ACM 23 (1976), 114–127

Vygen, J. [1997] : The two-dimensional weighted median problem. Zeitschrift für Angewandte Mathematik und Mechanik 77 (1997), Supplement, S433–S436

Woeginger, G.J. [2000] : When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS) ? INFORMS Journal on Computing 12 (2000), 57–74

Chapitre 18

Le problème du bin-packing

Supposons que nous ayons n objets, chacun d'une taille donnée, et des boîtes de même capacité. On veut ranger les objets dans les boîtes, en utilisant le moins de boîtes possible. Bien entendu, la taille totale des objets affectés à une boîte ne doit pas dépasser sa capacité.

Sans perte de généralité, on suppose que la capacité des boîtes est égale à 1. Alors le problème peut être formulé de la manière suivante :

PROBLÈME DU BIN-PACKING	
Instance	Une liste de nombres non négatifs $a_1, \ldots, a_n \leq 1$.
Tâche	Trouver un $k \in \mathbb{N}$ et une affectation $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ telle que $\sum_{i:f(i)=j} a_i \leq 1$ pour tout $j \in \{1, \ldots, k\}$ et telle que k soit minimum.

Peu de problèmes d'optimisation combinatoire ont un intérêt pratique aussi évident. Par exemple, la version la plus simple du problème de découpe est équivalente au PROBLÈME DU BIN-PACKING : étant donné des poutres de longueurs égales (disons 1 mètre) et des nombres $a_1, \ldots, a_n$, on veut couper en morceaux aussi peu de poutres que possible de façon à obtenir finalement des poutres de longueur $a_1, \ldots, a_n$.

Bien qu'une instance I soit une liste ordonnée de nombres qui peuvent apparaître plusieurs fois, on écrit $x \in I$ pour un élément de la liste I qui est égal à x. On note $|I|$ le nombre d'éléments de la liste I. Nous utiliserons aussi l'abréviation $\mathrm{SUM}(a_1, \ldots, a_n) := \sum_{i=1}^{n} a_i$. Cela constitue une borne inférieure évidente : l'inégalité $\lceil \mathrm{SUM}(I) \rceil \leq \mathrm{OPT}(I)$ est vérifiée par toute instance I.

Nous prouvons au paragraphe 18.1 que le PROBLÈME DU BIN-PACKING est fortement *NP*-difficile et présentons quelques algorithmes d'approximation simples. Nous verrons qu'aucun algorithme ne peut atteindre un rapport de performance meilleur que $\frac{3}{2}$ (à moins que $P = NP$). Cependant, on peut atteindre asymptotiquement un rapport de performance arbitrairement bon : aux paragraphes 18.2 et 18.3 nous décrivons un schéma d'approximation asymptotique entièrement polyno-

mial. Il nécessite l'utilisation de la MÉTHODE DES ELLIPSOÏDES et des résultats du chapitre 17.

18.1 Heuristiques gloutonnes

Nous analyserons dans ce paragraphe quelques heuristiques gloutonnes pour le PROBLÈME DU BIN-PACKING. On ne peut espérer l'existence d'un algorithme polynomial exact puisque le problème est *NP*-difficile :

Théorème 18.1. *Le problème suivant est NP-complet : étant donné une instance I du* PROBLÈME DU BIN-PACKING*, décider si I a une solution avec deux boîtes.*

Preuve. L'appartenance à *NP* est évidente. Nous montrons que le problème de décision précédent s'obtient par transformation du PROBLÈME DE LA PARTITION (qui est *NP*-complet : corollaire 15.28). Étant donné une instance $c_1, \ldots, c_n$ du PROBLÈME DE LA PARTITION, considérons l'instance $a_1, \ldots, a_n$ du PROBLÈME DU BIN-PACKING, où

$$a_i \;=\; \frac{2c_i}{\sum_{j=1}^{n} c_j}.$$

Évidemment deux boîtes suffisent si et seulement s'il existe un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$. □

Corollaire 18.2. *À moins que P = NP, il n'existe pas d'algorithme d'approximation de facteur ρ pour le* PROBLÈME DU BIN-PACKING *pout tout $\rho < \frac{3}{2}$.* □

Pour tout k fixé, il existe un algorithme pseudo-polynomial qui permet de décider, pour une instance I donnée, si k boîtes suffisent (exercice 1). Cependant, ce problème est en général fortement *NP*-complet :

Théorème 18.3. (Garey et Johnson [1975]) *Le problème suivant est fortement NP-complet : étant donné une instance I du* PROBLÈME DU BIN-PACKING *et un nombre B, décider si I peut être résolu avec B boîtes.*

Preuve. Transformation à partir du COUPLAGE 3-DIMENSIONNEL (théorème 15.26).

Étant donné une instance U, V, W, T du problème 3DM, on construit une instance I du PROBLÈME DU BIN-PACKING avec $4|T|$ objets. En fait, l'ensemble des objets est

$$S \;:=\; \bigcup_{t=(u,v,w)\in T} \{t, (u,t), (v,t), (w,t)\}.$$

Considérons $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$ et $W = \{w_1, \ldots, w_n\}$. Pour chaque $x \in U \,\dot\cup\, V \,\dot\cup\, W$, on choisit $t_x \in T$ tel que $(x, t_x) \in S$. Pour chaque $t = (u_i, v_j, w_k) \in T$, les tailles des objets sont maintenant définies de la façon suivante :

$$
\begin{array}{rl}
t & \text{est de taille } \frac{1}{C}(10N^4 + 8 - iN - jN^2 - kN^3) \\
(u_i, t) & \text{est de taille } \begin{cases} \frac{1}{C}(10N^4 + iN + 1) & \text{si } t = t_{u_i} \\ \frac{1}{C}(11N^4 + iN + 1) & \text{si } t \neq t_{u_i} \end{cases} \\
(v_j, t) & \text{est de taille } \begin{cases} \frac{1}{C}(10N^4 + jN^2 + 2) & \text{si } t = t_{v_j} \\ \frac{1}{C}(11N^4 + jN^2 + 2) & \text{si } t \neq t_{v_j} \end{cases} \\
(w_k, t) & \text{est de taille } \begin{cases} \frac{1}{C}(10N^4 + kN^3 + 4) & \text{si } t = t_{w_k} \\ \frac{1}{C}(8N^4 + kN^3 + 4) & \text{si } t \neq t_{w_k} \end{cases}
\end{array}
$$

où $N := 100n$ et $C := 40N^4 + 15$. Cela définit une instance $I = (a_1, \ldots, a_{4|T|})$ du PROBLÈME DU BIN-PACKING. Nous posons $B := |T|$ et affirmons que I a une solution avec au plus B boîtes si et seulement si l'instance initiale du problème 3DM est une instance-«oui», c.-à-d. s'il existe un sous-ensemble M de T avec $|M| = n$ tel que pour des éléments distincts $(u, v, w), (u', v', w') \in M$ on a $u \neq u'$, $v \neq v'$ et $w \neq w'$.

Supposons d'abord qu'il existe une telle solution M de l'instance du problème 3DM. Puisque l'existence d'une solution pour l'instance I avec B boîtes est indépendante du choix des t_x ($x \in U \cup V \cup W$), on peut les redéfinir de façon que $t_x \in M$ pour tout x. On range maintenant, pour chaque $t = (u, v, w) \in T$, $t, (u, t), (v, t), (w, t)$ dans une seule boîte. Cela fournit une solution avec $|T|$ boîtes.

Inversement, soit f une solution de I avec $B = |T|$ boîtes. Comme $\mathrm{SUM}(I) = |T|$, chaque boîte doit être complètement pleine. Puisque toutes les tailles des objets sont strictement comprises entre $\frac{1}{5}$ et $\frac{1}{3}$, chaque boîte doit contenir quatre objets.

Considérons une boîte $k \in \{1, \ldots, B\}$. Comme $C \sum_{i:f(i)=k} a_i = C \equiv 15 \pmod N$, la boîte doit contenir un $t = (u, v, w) \in T$, un $(u', t') \in U \times T$, un $(v', t'') \in V \times T$, et un $(w', t''') \in W \times T$. Comme $C \sum_{i:f(i)=k} a_i = C \equiv 15 \pmod{N^2}$, on a $u = u'$. De manière similaire, en considérant la somme modulo N^3 et modulo N^4, on obtient $v = v'$ et $w = w'$. De plus, soit $t' = t_u$ et $t'' = t_v$ et $t''' = t_w$ (cas 1) ou $t' \neq t_u$ et $t'' \neq t_v$ et $t''' \neq t_w$ (cas 2).

On définit M comme l'ensemble des $t \in T$ pour lesquels t est affecté à une boîte qui correspond au cas 1. Évidemment M est une solution de l'instance du problème 3DM.

Remarquons que tous les nombres de l'instance construite I du PROBLÈME DU BIN-PACKING sont «polynomialement» grands, plus précisément en $O(n^4)$. Puisque le problème 3DM est fortement *NP*-complet (théorème 15.26), le théorème est prouvé. □

Cette preuve est due à Papadimitriou [1994]. Même sous l'hypothèse $P \neq NP$, le résultat précédent n'exclut pas l'existence d'un algorithme d'approximation absolu, par exemple un algorithme qui nécessiterait au plus une boîte de plus que la solution optimale. Savoir si un tel algorithme existe est une question ouverte.

L'algorithme suivant, dit de la zone libre suivante ou de l'ajustement suivant (en anglais *next-fit*), est sans doute l'heuristique la plus simple pour le PROBLÈME DU BIN-PACKING :

ALGORITHME NEXT-FIT (NF)	
Input	Une instance $a_1, \ldots, a_n$ du PROBLÈME DU BIN-PACKING.
Output	Une solution (k, f).

① Poser $k := 1$ et $S := 0$.

② **For** $i := 1$ **to** n **do** :
  If $S + a_i > 1$ **then** poser $k := k + 1$ et $S := 0$.
  Poser $f(i) := k$ et $S := S + a_i$.

Notons $NF(I)$ le nombre k de boîtes que cet algorithme utilise pour l'instance I.

Théorème 18.4. *L'*ALGORITHME NEXT-FIT *s'exécute en temps $O(n)$. Pour toute instance $I = a_1, \ldots, a_n$, on a*

$$NF(I) \;\leq\; 2\lceil \mathrm{SUM}(I) \rceil - 1 \;\leq\; 2\,\mathrm{OPT}(I) - 1.$$

Preuve. La borne sur le temps de calcul est évidente. Soit $k := NF(I)$, et soit f l'affectation trouvée par l'ALGORITHME NEXT-FIT. Pour $j = 1, \ldots, \lfloor \frac{k}{2} \rfloor$, on a

$$\sum_{i : f(i) \in \{2j-1, 2j\}} a_i \;>\; 1.$$

En additionnant ces inégalités on obtient

$$\left\lfloor \frac{k}{2} \right\rfloor \;<\; \mathrm{SUM}(I).$$

Comme le terme de gauche est un entier, on conclut que

$$\frac{k-1}{2} \;\leq\; \left\lfloor \frac{k}{2} \right\rfloor \;\leq\; \lceil \mathrm{SUM}(I) \rceil - 1.$$

Cela prouve $k \leq 2\lceil \mathrm{SUM}(I) \rceil - 1$. La seconde inégalité est triviale. □

Les instances $2\epsilon, 1-\epsilon, 2\epsilon, 1-\epsilon, \ldots, 2\epsilon$ pour de très petits $\epsilon > 0$ montrent que cette borne est la meilleure possible. L'ALGORITHME NEXT-FIT est donc un algorithme d'approximation de facteur 2. Naturellement le rapport de performance s'améliore si les nombres mis en jeu sont petits :

Proposition 18.5. *Soit $0 < \gamma < 1$. Pour toute instance $I = a_1, \ldots, a_n$ telle que $a_i \leq \gamma$ pour tout $i \in \{1, \ldots, n\}$, on a*

$$NF(I) \;\leq\; \left\lceil \frac{\mathrm{SUM}(I)}{1-\gamma} \right\rceil.$$

Preuve. On a $\sum_{i:f(i)=j} a_i > 1-\gamma$ pour $j = 1, \ldots, NF(I)-1$. En ajoutant ces inégalités, on obtient $(NF(I)-1)(1-\gamma) < \mathrm{SUM}(I)$ et ainsi

$$NF(I) - 1 \;\leq\; \left\lceil \frac{\mathrm{SUM}(I)}{1-\gamma} \right\rceil - 1.$$

□

L'algorithme suivant, dit de la première zone libre ou du premier ajustement (en anglais *first-fit*), est une seconde approche possible pour construire un algorithme d'approximation efficace :

ALGORITHME FIRST-FIT (FF)	
Input	Une instance $a_1, \ldots, a_n$ du PROBLÈME DU BIN-PACKING.
Output	Une solution (k, f).

① **For** $i := 1$ **to** n **do** :

$$\text{Poser } f(i) := \min \left\{ j \in \mathbb{N} : \sum_{h<i:f(h)=j} a_h + a_i \leq 1 \right\}.$$

② Poser $k := \max_{i \in \{1,\ldots,n\}} f(i)$.

Bien entendu, l'ALGORITHME FIRST-FIT ne peut pas être plus mauvais que l'ALGORITHME NEXT-FIT. L'ALGORITHME FIRST-FIT est donc un autre algorithme d'approximation de facteur 2. Il est en fait meilleur :

Théorème 18.6. (Johnson *et al.* [1974], Garey *et al.* [1976]) *Pour toutes les instances I du* PROBLÈME DU BIN-PACKING*,*

$$FF(I) \;\leq\; \left\lceil \frac{17}{10}\,\mathrm{OPT}(I) \right\rceil.$$

De plus, il existe des instances I avec $\mathrm{OPT}(I)$ *arbitrairement grand et*

$$FF(I) \;\geq\; \frac{17}{10}(\mathrm{OPT}(I) - 1).$$

Nous omettons la preuve qui est assez compliquée. Pour de petites valeurs de $\mathrm{OPT}(I)$, la borne $FF(I) \leq \frac{7}{4}\,\mathrm{OPT}(I)$ obtenue par Simchi-Levi [1994] est meilleure.

La proposition 18.5 montre que l'ALGORITHME NEXT-FIT (et donc l'ALGORITHME FIRST-FIT) se comporte bien si les objets sont petits. Il est donc naturel de vouloir traiter les grands objets en premier. La modification suivante de l'ALGORITHME FIRST-FIT, appelé algorithme du premier ajustement décroissant (en anglais *first-fit decreasing*), examine les n nombres par ordre décroissant :

ALGORITHME FIRST-FIT-DÉCROISSANT (FFD)	
Input	Une instance $a_1, \ldots, a_n$ du PROBLÈME DU BIN-PACKING.
Output	Une solution (k, f).

① Trier les nombres de telle façon que $a_1 \geq a_2 \geq \ldots \geq a_n$.

② Appliquer l'ALGORITHME FIRST-FIT.

Théorème 18.7. (Simchi-Levi [1994]) *L'*ALGORITHME FIRST-FIT-DÉCROISSANT *fournit une approximation de facteur* $\frac{3}{2}$ *pour le* PROBLÈME DU BIN-PACKING.

Preuve. Soit I une instance et soit $k := FFD(I)$. Considérons la j-ième boîte pour $j := \lceil \frac{2}{3}k \rceil$. Si elle contient un objet de taille $> \frac{1}{2}$, alors chaque boîte d'indice plus petit n'avait pas assez de place pour cet objet et contenait donc déjà un objet. Comme les objets sont considérés par ordre décroissant, il y a au moins j objets de taille $> \frac{1}{2}$. Ainsi $\mathrm{OPT}(I) \geq j \geq \frac{2}{3}k$.

Sinon la j-ième boîte, et donc chaque boîte d'indice plus grand, ne contient pas d'objet de taille $> \frac{1}{2}$. Ainsi les boîtes $j, j+1, \ldots, k$ contiennent au moins $2(k-j)+1$ objets, aucun d'entre eux ne pouvant être contenu par les boîtes $1, \ldots, j-1$. Ainsi $\mathrm{SUM}(I) > \min\{j-1, 2(k-j)+1\} \geq \min\{\lceil \frac{2}{3}k \rceil - 1, 2(k-(\frac{2}{3}k+\frac{2}{3}))+1\} = \lceil \frac{2}{3}k \rceil - 1$ et $\mathrm{OPT}(I) \geq \mathrm{SUM}(I) > \lceil \frac{2}{3}k \rceil - 1$, c.-à-d. $\mathrm{OPT}(I) \geq \lceil \frac{2}{3}k \rceil$. □

D'après le corollaire 18.2, on ne peut trouver un meilleur algorithme d'approximation (pour l'algorithme FFD, considérer l'instance 0,4, 0,4, 0,3, 0,3, 0,3, 0,3). Cependant, la garantie de performance asymptotique est meilleure : Johnson [1973] a prouvé que $FFD(I) \leq \frac{11}{9}\,\mathrm{OPT}(I) + 4$ pour toutes les instances I (voir également Johnson [1974]). Baker [1985] a donné une preuve plus simple montrant que $FFD(I) \leq \frac{11}{9}\,\mathrm{OPT}(I) + 3$. Le meilleur résultat connu est le suivant :

Théorème 18.8. (Yue [1990]) *Pour toutes les instances I du* PROBLÈME DU BIN-PACKING,

$$FFD(I) \;\leq\; \frac{11}{9}\,\mathrm{OPT}(I) + 1.$$

La preuve de Yue est plus courte que les précédentes, mais encore trop compliquée pour être présentée ici. En revanche, nous présentons une classe d'instances I telle que $\mathrm{OPT}(I)$ est arbitrairement grand et $FFD(I) = \frac{11}{9}\,\mathrm{OPT}(I)$. (Cet exemple est dû à Garey et Johnson [1979].)

Soit $\epsilon > 0$ suffisamment petit et $I = \{a_1, \ldots, a_{30m}\}$ tel que

$$a_i = \begin{cases} \frac{1}{2} + \epsilon & \text{si } 1 \leq i \leq 6m, \\ \frac{1}{4} + 2\epsilon & \text{si } 6m < i \leq 12m, \\ \frac{1}{4} + \epsilon & \text{si } 12m < i \leq 18m, \\ \frac{1}{4} - 2\epsilon & \text{si } 18m < i \leq 30m. \end{cases}$$

La solution optimale est constituée de

$$\begin{array}{ll} 6m \text{ boîtes contenant} & \frac{1}{2} + \epsilon,\ \frac{1}{4} + \epsilon,\ \frac{1}{4} - 2\epsilon, \\ 3m \text{ boîtes contenant} & \frac{1}{4} + 2\epsilon,\ \frac{1}{4} + 2\epsilon, \frac{1}{4} - 2\epsilon,\ \frac{1}{4} - 2\epsilon. \end{array}$$

La solution renvoyée par l'algorithme FFD est constituée de

$$\begin{array}{ll} 6m \text{ boîtes contenant} & \frac{1}{2} + \epsilon,\ \frac{1}{4} + 2\epsilon, \\ 2m \text{ boîtes contenant} & \frac{1}{4} + \epsilon,\ \frac{1}{4} + \epsilon,\ \frac{1}{4} + \epsilon, \\ 3m \text{ boîtes contenant} & \frac{1}{4} - 2\epsilon,\ \frac{1}{4} - 2\epsilon,\ \frac{1}{4} - 2\epsilon,\ \frac{1}{4} - 2\epsilon. \end{array}$$

Donc $\mathrm{OPT}(I) = 9m$ et $FFD(I) = 11m$.

Il existe d'autres algorithmes pour le PROBLÈME DU BIN-PACKING, certains d'entre eux ayant un rapport de performance asymptotique meilleur que $\frac{11}{9}$. Dans le paragraphe suivant, on montre que l'on peut atteindre un rapport de performance asymptotique arbitrairement proche de 1.

Dans certaines applications, on doit ranger les objets dans l'ordre dans lequel ils arrivent sans connaître les objets suivants. Les algorithmes qui n'utilisent pas d'information liée aux objets suivants sont appelés des algorithmes en ligne (en anglais *online algorithm*). Par exemple, les algorithmes NEXT-FIT et FIRST-FIT sont des algorithmes en ligne, mais l'ALGORITHME FIRST-FIT-DÉCROISSANT n'est pas un algorithme en ligne. Le meilleur algorithme en ligne connu pour le PROBLÈME DU BIN-PACKING a un rapport de performance asymptotique égal à $1,59$ (Seiden [2002]). D'autre part, Van Vliet [1992] a prouvé qu'il n'existe pas d'algorithme en ligne d'approximation asymptotique de facteur $1,54$ pour le PROBLÈME DU BIN-PACKING. La preuve d'un résultat semblable, mais avec une borne inférieure plus faible est le sujet de l'exercice 5.

18.2 Un schéma d'approximation asymptotique

Dans cette section, nous montrons que, pour tout $\epsilon > 0$, il existe un algorithme linéaire qui garantit de trouver une solution avec au plus $(1+\epsilon)\,\mathrm{OPT}(I) + \frac{1}{\epsilon^2}$ boîtes.

Nous commençons par étudier des instances qui ne contiennent pas trop de nombres différents. Nous désignons les différents nombres de notre instance I par $s_1, \ldots, s_m$. Supposons que I contienne exactement b_i copies de s_i $(i = 1, \ldots, m)$.

Notons $T_1, \ldots, T_N$ toutes les manières de remplir une seule boîte :

$$\{T_1, \ldots, T_N\} := \left\{ (k_1, \ldots, k_m) \in \mathbb{Z}_+^m : \sum_{i=1}^m k_i s_i \leq 1 \right\}.$$

On note $T_j = (t_{j1}, \ldots, t_{jm})$. Alors notre PROBLÈME DU BIN-PACKING est équivalent au PL en nombres entiers suivant (dû à Eisemann [1957]) :

$$\begin{array}{lll} \min & \displaystyle\sum_{j=1}^N x_j & \\ \text{s.c.} & \displaystyle\sum_{j=1}^N t_{ji} x_j \geq b_i & (i = 1, \ldots, m) \\ & x_j \in \mathbb{Z}_+ & (j = 1, \ldots, N). \end{array} \tag{18.1}$$

Nous voulons en fait avoir $\sum_{j=1}^N t_{ji} x_j = b_i$, mais relâcher cette contrainte ne change rien. La relaxation linéaire de (18.1) s'écrit :

$$\begin{array}{lll} \min & \displaystyle\sum_{j=1}^N x_j & \\ \text{s.c.} & \displaystyle\sum_{j=1}^N t_{ji} x_j \geq b_i & (i = 1, \ldots, m) \\ & x_j \geq 0 & (j = 1, \ldots, N). \end{array} \tag{18.2}$$

Le théorème suivant dit qu'en arrondissant une solution de la relaxation linéaire (18.2) on obtient une solution de (18.1), c.-à-d. du PROBLÈME DU BIN-PACKING, qui n'est pas tellement plus mauvaise :

Théorème 18.9. (Fernandez de la Vega et Lueker [1981]) *Soit I une instance du* PROBLÈME DU BIN-PACKING *avec seulement m nombres différents. Soit x une solution réalisable (pas nécessairement optimale) de (18.2) avec au plus m composantes non nulles. Alors une solution du* PROBLÈME DU BIN-PACKING *avec au plus $\left\lceil \sum_{j=1}^N x_j \right\rceil + \left\lfloor \frac{m-1}{2} \right\rfloor$ boîtes peut être trouvée en temps $O(|I|)$.*

Preuve. Considérons $\lfloor x \rfloor$, qui est obtenu de x en arrondissant chaque composante à l'entier inférieur. $\lfloor x \rfloor$ ne correspond généralement pas à un rangement complet de l'instance I (il se peut que certains objets soient rangés plus de fois que nécessaire, mais cela n'a pas d'importance). Les objets restants forment une instance I'. Observons que

$$\mathrm{SUM}(I') \;\le\; \sum_{j=1}^{N}(x_j - \lfloor x_j \rfloor)\sum_{i=1}^{m} t_{ji}s_i \;\le\; \sum_{j=1}^{N} x_j - \sum_{j=1}^{N}\lfloor x_j \rfloor.$$

Il suffit donc de ranger I' dans au plus $\lceil \mathrm{SUM}(I') \rceil + \left\lfloor \frac{m-1}{2} \right\rfloor$ boîtes, car alors le nombre total de boîtes utilisées ne dépasse pas

$$\sum_{j=1}^{N}\lfloor x_j \rfloor + \lceil \mathrm{SUM}(I') \rceil + \left\lfloor \frac{m-1}{2} \right\rfloor \;\le\; \left\lceil \sum_{j=1}^{N} x_j \right\rceil + \left\lfloor \frac{m-1}{2} \right\rfloor.$$

On considère deux méthodes de rangement pour I'. Tout d'abord, le vecteur $\lceil x \rceil - \lfloor x \rfloor$ correspond à un rangement qui place au moins les éléments de I' dans des boîtes. Le nombre de boîtes utilisées est au plus égal à m puisque x a au plus m composantes non nulles. Ensuite, on peut aussi obtenir un rangement de I' utilisant au plus $2\lceil \mathrm{SUM}(I') \rceil - 1$ boîtes en appliquant l'ALGORITHME NEXT-FIT (théorème 18.4). Ces deux rangements peuvent être obtenus en temps linéaire.

Le meilleur de ces deux rangements utilise au plus $\min\{m,\, 2\lceil \mathrm{SUM}(I') \rceil - 1\} \le \lceil \mathrm{SUM}(I') \rceil + \frac{m-1}{2}$ boîtes. Le théorème est ainsi prouvé. □

Corollaire 18.10. (Fernandez de la Vega et Lueker [1981]) *Soient m et $\gamma > 0$ des constantes fixées. Soit I une instance du* PROBLÈME DU BIN-PACKING *avec seulement m nombres différents, dont aucun n'est inférieur à γ. On peut alors trouver une solution avec au plus $\mathrm{OPT}(I) + \left\lfloor \frac{m-1}{2} \right\rfloor$ boîtes en temps $O(|I|)$.*

Preuve. À l'aide de l'ALGORITHME DU SIMPLEXE (théorème 3.14), on peut trouver une solution optimale de base x^* de (18.2), c.-à-d. un sommet du polyèdre. Puisque tout sommet satisfait N des contraintes à égalité (proposition 3.9), x^* a au plus m composantes non nulles.

Le temps nécessaire pour déterminer x^* dépend seulement de m et N. Observons que $N \le (m+1)^{\frac{1}{\gamma}}$, car il ne peut y avoir qu'au plus $\frac{1}{\gamma}$ éléments dans chaque boîte. x^* peut donc être trouvé en temps constant.

Comme $\left\lceil \sum_{j=1}^{N} x_j^* \right\rceil \le \mathrm{OPT}(I)$, une application du théorème 18.9 complète la preuve. □

Utiliser la MÉTHODE DES ELLIPSOÏDES (théorème 4.18) mène au même résultat. Celui-ci n'est pas le meilleur possible : on peut même déterminer une solution optimale exacte en temps polynomial pour m et γ fixés, puisque le problème de PROGRAMMATION EN NOMBRES ENTIERS avec un nombre constant de variables peut être résolu en temps polynomial (Lenstra [1983]). Cependant, cela ne nous aiderait pas beaucoup. Nous appliquerons de nouveau le théorème 18.9 au paragraphe suivant et obtiendrons la même garantie de performance en temps polynomial même si m et γ ne sont pas fixés (dans la preuve du théorème 18.14).

Nous pouvons maintenant présenter l'algorithme de Fernandez de la Vega et Lueker [1981]. Il procède de la manière suivante : on partitionne d'abord les n nombres en $m+2$ groupes selon leur taille. On range d'abord le groupe contenant

les plus grands nombres en utilisant une boîte pour chaque nombre. On range ensuite les m groupes intermédiaires en arrondissant d'abord la taille de chaque nombre au plus grand nombre de son groupe et en appliquant ensuite le corollaire 18.10. On range finalement le groupe contenant les plus petits nombres.

ALGORITHME DE FERNANDEZ-DE-LA-VEGA-LUEKER

Input Une instance $I = a_1, \ldots, a_n$ du PROBLÈME DU BIN-PACKING. Un nombre $\epsilon > 0$.

Output Une solution (k, f) pour I.

① Poser $\gamma := \frac{\epsilon}{\epsilon+1}$ et $h := \lceil \epsilon \ \mathrm{SUM}(I) \rceil$.

② Soit $I_1 = L, M, R$ un réarrangement de la liste I, où $M = K_0, y_1, K_1, y_2, \ldots, K_{m-1}, y_m$ et $L, K_0, K_1, \ldots, K_{m-1}$ et R sont encore des listes, telles que les propriétés suivantes soient vérifiées :
   (a) Pour tout $x \in L : x < \gamma$.
   (b) Pour tout $x \in K_0 : \gamma \leq x \leq y_1$.
   (c) Pour tout $x \in K_i : y_i \leq x \leq y_{i+1} \quad (i = 1, \ldots, m-1)$.
   (d) Pour tout $x \in R : y_m \leq x$.
   (e) $|K_1| = \cdots = |K_{m-1}| = |R| = h - 1$ et $|K_0| \leq h - 1$.
   (k, f) est maintenant déterminé à l'aide des trois étapes de rangement suivantes :

③ Trouver un rangement S_R de R utilisant $|R|$ boîtes.

④ Considérons l'instance Q constituée des nombres $y_1, y_2, \ldots, y_m$, chacun apparaissant h fois. Trouver un rangement S_Q de Q utilisant au plus $\frac{m+1}{2}$ boîtes de plus que nécessaire (en utilisant le corollaire 18.10). Transformer S_Q en un rangement S_M de M.

⑤ Tant qu'une boîte de S_R ou S_M a un espace libre au moins égal à γ, la remplir avec des éléments de L. Finalement, trouver un rangement du reste de L en utilisant l'ALGORITHME NEXT-FIT.

À l'étape ④, nous avons utilisé une borne légèrement plus faible que celle obtenue au corollaire 18.10. Cela n'a pas d'importance ici et nous aurons besoin de la forme précédente au paragraphe 18.3. L'algorithme précédent est un schéma d'approximation asymptotique. Plus précisément :

Théorème 18.11. (Fernandez de la Vega et Lueker [1981]) *Pour tout* $0 < \epsilon \leq \frac{1}{2}$ *et pour toute instance* I *du* PROBLÈME DU BIN-PACKING, *l'*ALGORITHME DE FERNANDEZ-DE-LA-VEGA-LUEKER *renvoie une solution utilisant au plus* $(1 + \epsilon)\,\mathrm{OPT}(I) + \frac{1}{\epsilon^2}$ *boîtes. Le temps de calcul est* $O(n\frac{1}{\epsilon^2})$ *plus le temps nécessaire pour résoudre (18.2). Pour* ϵ *fixé, le temps de calcul est* $O(n)$.

Preuve. À l'étape ②, on détermine d'abord L en temps $O(n)$. Alors on pose $m := \left\lfloor \frac{|I|-|L|}{h} \right\rfloor$. Puisque $\gamma(|I| - |L|) \leq \mathrm{SUM}(I)$, on a

$$m \;\leq\; \frac{|I|-|L|}{h} \;\leq\; \frac{|I|-|L|}{\epsilon\,\mathrm{SUM}(I)} \;\leq\; \frac{1}{\gamma\epsilon} \;=\; \frac{\epsilon+1}{\epsilon^2}.$$

On sait que y_i doit être le $(|I|+1-(m-i+1)h)$-ième plus petit élément $(i=1,\ldots,m)$. Donc, d'après le corollaire 17.4, on peut trouver chaque y_i en temps $O(n)$. On détermine finalement $K_0, K_1, \ldots, K_{m-1}, R$, chacun en temps $O(n)$. L'étape ② peut donc être réalisée en temps $O(mn)$. Remarquons que $m = O(\frac{1}{\epsilon^2})$.

Les étapes ③, ④ et ⑤ – excepté la solution de (18.2) – peuvent facilement être implémentées de telle façon qu'elles s'exécutent en temps $O(n)$. Pour ϵ fixé, (18.2) peut aussi être résolu de manière optimale en temps $O(n)$ (corollaire 18.10).

Nous prouvons maintenant la garantie de performance. Soit k le nombre de boîtes que l'algorithme utilise. On note $|S_R|$ et $|S_M|$ le nombre de boîtes utilisées pour ranger R et M, respectivement.

On a

$$|S_R| \;\leq\; |R| \;=\; h-1 \;<\; \epsilon\,\mathrm{SUM}(I) \;\leq\; \epsilon\,\mathrm{OPT}(I).$$

Ensuite, observons que $\mathrm{OPT}(Q) \leq \mathrm{OPT}(I)$: le i-ième plus grand élément de I est supérieur ou égal au i-ième plus grand élément de Q pour tout $i = 1, \ldots, hm$. Ainsi, d'après l'étape ④ (corollaire 18.10), on a

$$|S_M| \;=\; |S_Q| \;\leq\; \mathrm{OPT}(Q) + \frac{m+1}{2} \;\leq\; \mathrm{OPT}(I) + \frac{m+1}{2}.$$

À l'étape ⑤ on peut ranger certains éléments de L dans des boîtes de S_R et S_M. Soit L' la liste des éléments restants de L.

Cas 1 : L' est non vide. Alors la taille totale des éléments dans chaque boîte, excepté éventuellement la dernière, dépasse $1-\gamma$, on a donc $(1-\gamma)(k-1) < \mathrm{SUM}(I) \leq \mathrm{OPT}(I)$. On en conclut que

$$k \;\leq\; \frac{1}{1-\gamma}\,\mathrm{OPT}(I) + 1 \;=\; (1+\epsilon)\,\mathrm{OPT}(I) + 1.$$

Cas 2 : L' est vide. Alors

$$\begin{aligned} k &\leq |S_R| + |S_M| \\ &< \epsilon\,\mathrm{OPT}(I) + \mathrm{OPT}(I) + \frac{m+1}{2} \\ &\leq (1+\epsilon)\,\mathrm{OPT}(I) + \frac{\epsilon+1+\epsilon^2}{2\epsilon^2} \\ &\leq (1+\epsilon)\,\mathrm{OPT}(I) + \frac{1}{\epsilon^2}, \end{aligned}$$

car $\epsilon \leq \frac{1}{2}$. □

Bien entendu, le temps de calcul augmente exponentiellement en $\frac{1}{\epsilon}$. Cependant, Karmarkar et Karp ont montré comment obtenir un schéma d'approximation asymptotique entièrement polynomial. Cela est le sujet du paragraphe suivant.

18.3 Algorithme de Karmarkar-Karp

L'algorithme de Karmarkar et Karp [1982] fonctionne de la même façon que l'algorithme du paragraphe précédent, mais au lieu de résoudre optimalement la relaxation linéaire (18.2) comme au corollaire 18.10, ils la résolvent à une erreur constante près.

Le fait que le nombre de variables augmente exponentiellement en $\frac{1}{\epsilon}$ ne nous empêchera pas de résoudre le PL : Gilmore et Gomory [1961] ont développé la technique de génération de colonnes et ont obtenu une variante de l'ALGORITHME DU SIMPLEXE qui permet de résoudre (18.2) assez efficacement en pratique. Des idées similaires mènent à un algorithme efficace théoriquement si on utilise à la place l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER.

Dans les deux approches mentionnées précédemment, le PL dual joue un rôle majeur. Le dual de (18.2) correspond au PL :

$$\begin{array}{lll} \max & yb & \\ \text{s.c.} & \sum_{i=1}^{m} t_{ji} y_i \leq 1 & (j = 1, \ldots, N) \\ & y_i \geq 0 & (i = 1, \ldots, m). \end{array} \tag{18.3}$$

Il a seulement m variables, mais un nombre exponentiel de contraintes. Cependant, le nombre de contraintes n'a pas d'importance du moment que l'on peut résoudre le PROBLÈME DE SÉPARATION en temps polynomial. Il apparaîtra que le PROBLÈME DE SÉPARATION est équivalent au PROBLÈME DU SAC À DOS. Puisque l'on peut résoudre le PROBLÈME DU SAC À DOS avec une erreur arbitrairement petite, on peut aussi résoudre le PROBLÈME DE SÉPARATION FAIBLE en temps polynomial. Cette idée nous permet de prouver le résultat suivant :

Lemme 18.12. (Karmarkar et Karp [1982]) *Soit I une instance du* PROBLÈME DU BIN-PACKING *avec seulement m nombres différents, aucun d'entre eux n'étant inférieur à γ. Soit $\delta > 0$. Alors il existe une solution réalisable y^* du PL dual (18.3), qui ne diffère de l'optimum que d'au plus δ et qui peut être trouvée en un temps $O\left(m^6 \log^2 \frac{mn}{\gamma\delta} + \frac{m^5 n}{\delta} \log \frac{mn}{\gamma\delta}\right)$.*

Preuve. On peut supposer que $\delta = \frac{1}{p}$ pour un certain entier naturel p. On applique l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER (théorème 4.19). Soit $\mathcal{D}$ le polyèdre associé au PL (18.3). On a

$$B\left(x_0, \frac{\gamma}{2}\right) \subseteq [0,\gamma]^m \subseteq \mathcal{D} \subseteq [0,1]^m \subseteq B(x_0, \sqrt{m}),$$

où x_0 est le vecteur dont toutes les composantes sont égales à $\frac{\gamma}{2}$.

Nous allons prouver que nous pouvons résoudre le PROBLÈME DE SÉPARATION FAIBLE pour (18.3), c.-à-d. avec $\mathcal{D}$, b et $\frac{\delta}{2}$ en temps $O\left(\frac{nm}{\delta}\right)$, indépendamment de la

taille du vecteur y. D'après le théorème 4.19, cela implique que le PROBLÈME D'OPTIMISATION FAIBLE peut être résolu en temps $O\left(m^6 \log^2 \frac{m||b||}{\gamma\delta} + \frac{m^5 n}{\delta} \log \frac{m||b||}{\gamma\delta}\right)$, ce qui prouve le lemme puisque $||b|| \leq n$.

Pour montrer comment résoudre le PROBLÈME DE SÉPARATION FAIBLE, considérons $y \in \mathbb{Q}^m$. On peut supposer $0 \leq y \leq 1$ puisque sinon la tâche est triviale. Observons maintenant que y est réalisable si et seulement si

$$\max\{yx : x \in \mathbb{Z}_+^m,\, xs \leq 1\} \;\leq\; 1, \tag{18.4}$$

où $s = (s_1, \ldots, s_m)$ est le vecteur des tailles des objets.

(18.4) est un problème du type PROBLÈME DU SAC À DOS, on ne peut donc pas espérer le résoudre exactement. Mais cela n'est pas nécessaire, puisque le PROBLÈME DE SÉPARATION FAIBLE ne requiert qu'une solution approchée.

Définissons $y' := \lfloor \frac{2n}{\delta} y \rfloor$ (l'arrondi est pris composante par composante). Le problème

$$\max\{y'x : x \in \mathbb{Z}_+^m,\, xs \leq 1\} \tag{18.5}$$

peut être résolu optimalement à l'aide de la programmation dynamique, de façon très similaire à l'ALGORITHME DE PROGRAMMATION DYNAMIQUE POUR LE PROBLÈME DU SAC À DOS du paragraphe 17.2 (voir exercice 6 du chapitre 17) : définissons $F(0) := 0$ et

$$F(k) \;:=\; \min\{F(k - y_i') + s_i : i \in \{1, \ldots, m\},\, y_i' \leq k\}$$

pour $k = 1, \ldots, \frac{4n}{\delta}$. $F(k)$ est la taille minimum d'un ensemble d'objets de coût total égal à k (respectivement à y').

Le maximum dans (18.5) est alors inférieur ou égal à $\frac{2n}{\delta}$ si et seulement si $F(k) > 1$ pour tout $k \in \{\frac{2n}{\delta} + 1, \ldots, \frac{4n}{\delta}\}$. Le temps total nécessaire pour décider cela est $O\left(\frac{mn}{\delta}\right)$. Il y a deux cas :

Cas 1 : le maximum dans (18.5) est inférieur ou égal à $\frac{2n}{\delta}$. Alors $\frac{\delta}{2n} y'$ est une solution réalisable de (18.3). De plus, $by - b\frac{\delta}{2n} y' \leq b\frac{\delta}{2n}\mathbb{1} = \frac{\delta}{2}$. La tâche du PROBLÈME DE SÉPARATION FAIBLE est réalisée.

Cas 2 : il existe un $x \in \mathbb{Z}_+^m$ tel que $xs \leq 1$ et $y'x > \frac{2n}{\delta}$. Un tel x peut être facilement calculé à partir des nombres $F(k)$ en temps $O\left(\frac{mn}{\delta}\right)$. On a $yx \geq \frac{\delta}{2n} y'x > 1$. Ainsi x correspond à une configuration de boîtes qui prouve que y n'est pas réalisable. Puisque l'on a $zx \leq 1$ pour tout $z \in \mathcal{D}$, c'est un hyperplan séparateur, et on a terminé. □

Lemme 18.13. (Karmarkar et Karp [1982]) *Soit I une instance du* PROBLÈME DU BIN-PACKING *avec seulement m nombres différents, aucun d'entre eux n'étant inférieur à γ. Soit $\delta > 0$. Alors une solution réalisable x du PL primal (18.2), qui ne diffère de l'optimum que d'au plus δ et qui a au plus m composantes non nulles, peut être trouvée en temps polynomial par rapport à n, $\frac{1}{\delta}$ et $\frac{1}{\gamma}$.*

Preuve. On résout d'abord approximativement le PL dual (18.3), en utilisant le lemme 18.12. On obtient un vecteur y^* tel que $y^*b \geq \mathrm{OPT}(18.3) - \delta$. Soient

maintenant $T_{k_1}, \ldots, T_{k_{N'}}$ les configurations de boîtes qui correspondent aux hyperplans séparateurs obtenus dans le cas 2 de la preuve précédente, plus les vecteurs unités (qui correspondent aux configurations de boîtes qui ne contiennent qu'un seul élément). Remarquons que N' est borné par le nombre d'itérations dans l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER (théorème 4.19), donc $N' = O\left(m^2 \log \frac{mn}{\gamma\delta}\right)$.

Considérons le PL

$$\begin{array}{lll} \max & yb & \\ \text{s.c.} & \sum_{i=1}^{m} t_{k_j i} y_i \leq 1 & (j = 1, \ldots, N') \\ & y_i \geq 0 & (i = 1, \ldots, m). \end{array} \tag{18.6}$$

Observons que la procédure précédente pour (18.3) (dans la preuve du lemme 18.12) est aussi une application valide de l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER pour (18.6) : l'oracle pour le PROBLÈME DE SÉPARATION FAIBLE peut toujours donner la même réponse que ci-dessus. Ainsi on a $y^*b \geq \mathrm{OPT}(18.6) - \delta$. Considérons

$$\begin{array}{lll} \min & \sum_{j=1}^{N'} x_{k_j} & \\ \text{s.c.} & \sum_{j=1}^{N'} t_{k_j i} x_{k_j} \geq b_i & (i = 1, \ldots, m) \\ & x_{k_j} \geq 0 & (j = 1, \ldots, N') \end{array} \tag{18.7}$$

qui est le dual de (18.6). Le PL (18.7) s'obtient à partir de (18.2) en éliminant les variables x_j pour $j \in \{1, \ldots, N\} \setminus \{k_1, \ldots, k_{N'}\}$ (en les forçant à être nulles). Autrement dit, seulement N' des N configurations de boîtes peuvent être utilisées.

On a

$$\mathrm{OPT}(18.7) - \delta \;=\; \mathrm{OPT}(18.6) - \delta \;\leq\; y^*b \;\leq\; \mathrm{OPT}(18.3) \;=\; \mathrm{OPT}(18.2).$$

Il suffit donc de résoudre (18.7). Mais (18.7) est un PL de taille polynomiale : il a N' variables et m contraintes ; aucune des entrées de la matrice n'est supérieure à $\frac{1}{\gamma}$ et aucune des entrées du terme de droite n'est supérieure à n. Donc, d'après le théorème de Khachiyan 4.18, il peut être résolu en temps polynomial. On obtient une solution optimale de base x (x est un sommet du polyèdre, donc x a au plus m composantes non nulles). □

Appliquons alors l'ALGORITHME DE FERNANDEZ-DE-LA-VEGA-LUEKER avec juste une modification : on remplace la solution exacte de (18.2) par une application du lemme 18.13. Nous résumons :

Théorème 18.14. (Karmarkar et Karp [1982]) *Il existe un schéma d'approximation asymptotique entièrement polynomial pour le* PROBLÈME DU BIN-PACKING.

Preuve. On applique le lemme 18.13 avec $\delta = 1$ pour obtenir une solution optimale x de (18.7) avec au plus m composantes non nulles. On a $\mathbb{1}x \leq \mathrm{OPT}(18.2) + 1$. Une application du théorème 18.9 fournit une solution entière utilisant au plus $\lceil \mathrm{OPT}(18.2) \rceil + 1 + \frac{m-1}{2}$ boîtes, comme requis à l'étape ④ de l'ALGORITHME DE FERNANDEZ-DE-LA-VEGA-LUEKER.

Donc le résultat du théorème 18.11 reste valide. Puisque $m \leq \frac{2}{\epsilon^2}$ et $\frac{1}{\gamma} \leq \frac{2}{\epsilon}$ (on peut supposer $\epsilon \leq 1$), le temps de calcul pour trouver x est polynomial par rapport à n et $\frac{1}{\epsilon}$. □

Le temps de calcul obtenu de cette façon est pire que $O\left(\epsilon^{-40}\right)$ et est donc totalement inacceptable en pratique. Karmarkar et Karp [1982] ont montré comment réduire le nombre de variables de (18.7) à m (tout en ne changeant que très légèrement la valeur optimale) et ont amélioré ainsi le temps de calcul (voir exercice 9). Plotkin, Shmoys et Tardos [1995] ont obtenu un temps de calcul en $O(n \log \epsilon^{-1} + \epsilon^{-6} \log \epsilon^{-1})$.

De nombreuses généralisations ont été étudiées. En dimension deux, quand on cherche à ranger un ensemble donné de rectangles parallèles dans un nombre minimum de carrés unitaires sans utiliser de rotation, le PROBLÈME DU BIN-PACKING n'a pas de schéma d'approximation asymptotique, à moins que *P = NP* (Bansal *et al.* [2006]). Voir Caprara [2002] et Zhang [2005] pour des résultats connexes.

Exercices

1. Soit k fixé. Décrire un algorithme pseudo-polynomial qui – pour une instance donnée I du PROBLÈME DU BIN-PACKING – trouve une solution pour cette instance n'utilisant pas plus de k boîtes ou conclut qu'une telle solution n'existe pas.
2. Supposons que pour une instance $a_1, \ldots, a_n$ du PROBLÈME DU BIN-PACKING on ait $a_i > \frac{1}{3}$ pour chaque i. Réduire le problème au PROBLÈME DU COUPLAGE MAXIMUM. Montrer ensuite comment le résoudre en temps $O(n \log n)$.
3. Trouver une instance I du PROBLÈME DU BIN-PACKING, telle que $FF(I) = 17$ alors que $\mathrm{OPT}(I) = 10$.
4. Implémenter l'ALGORITHME FIRST-FIT et l'ALGORITHME FIRST-FIT-DÉCROISSANT de telle façon qu'ils s'exécutent en temps $O(n \log n)$.
5. Montrer qu'il n'existe pas d'algorithme en ligne d'approximation de facteur $\frac{4}{3}$ pour le PROBLÈME DU BIN-PACKING à moins que *P = NP*.
   Indication : considérer la liste constituée de n éléments de taille $\frac{1}{2} - \epsilon$ suivis de n éléments de taille $\frac{1}{2} + \epsilon$.
6. Montrer que l'étape ② de l'ALGORITHME DE FERNANDEZ-DE-LA-VEGA-LUEKER peut être implémentée de telle façon à s'exécuter en temps $O\left(n \log \frac{1}{\epsilon}\right)$.

* 7. Prouver que, pour tout $\epsilon > 0$, il existe un algorithme polynomial qui, pour toute instance $I = (a_1, \ldots, a_n)$ du PROBLÈME DU BIN-PACKING, trouve un rangement qui utilise le nombre optimum de boîtes, mais qui peut violer les contraintes de capacité de ϵ, c.-à-d. une affectation $f : \{1, \ldots, n\} \to \{1, \ldots, \mathrm{OPT}(I)\}$ telle que $\sum_{f(i)=j} a_i \leq 1+\epsilon$ pour tout $j \in \{1, \ldots, \mathrm{OPT}(I)\}$.
Indication : utiliser les idées du paragraphe 18.2.
(Hochbaum et Shmoys [1987])

8. Considérer le PROBLÈME D'ORDONNANCEMENT MULTIPROCESSEUR suivant. Étant donné un ensemble fini A de tâches, un nombre positif $t(a)$ pour chaque $a \in A$ (le temps de traitement) et un nombre m de processeurs. Trouver une partition $A = A_1 \,\dot\cup\, A_2 \,\dot\cup\, \cdots \,\dot\cup\, A_m$ de A en m ensembles deux à deux disjoints telle que $\max_{i=1}^{m} \sum_{a \in A_i} t(a)$ soit minimum.

   (a) Montrer que ce problème est fortement *NP*-difficile.

   (b) Montrer qu'un algorithme glouton qui affecte successivement les tâches (dans un ordre arbitraire) à la machine la moins utilisée constitue un algorithme d'approximation de facteur 2.

   (c) Montrer que, pour chaque valeur de m fixée, le problème a un schéma d'approximation entièrement polynomial.
   (Horowitz et Sahni [1976])

* (d) Utiliser l'exercice 7 pour montrer que le PROBLÈME D'ORDONNANCEMENT MULTIPROCESSEUR a un schéma d'approximation.
   (Hochbaum et Shmoys [1987])

Remarque : ce problème a été le sujet du premier article sur les algorithmes d'approximation (Graham [1966]). De nombreuses variantes du problème d'ordonnancement ont été étudiées ; voir par exemple Graham *et al.* [1979] ou Lawler *et al.* [1993].

* 9. Considérons le PL (18.6) de la preuve du lemme 18.13. On peut omettre presque toutes les contraintes sauf m d'entre elles sans que cela change la valeur optimale du PL. Nous ne sommes pas capables de trouver ces m contraintes en temps polynomial, mais nous pouvons trouver m contraintes telles que la suppression de toutes les autres n'augmente pas beaucoup la valeur optimale (par exemple d'au plus un). Comment ?
Indication : soit $D^{(0)}$ le PL (18.6), construire ensuite itérativement les PLs $D^{(1)}, D^{(2)}, \ldots$ en supprimant de plus en plus de contraintes. À chaque itération, une solution $y^{(i)}$ de $D^{(i)}$ telle que $by^{(i)} \geq \mathrm{OPT}\left(D^{(i)}\right) - \delta$ est donnée. L'ensemble des contraintes est partitionné en $m+1$ ensembles ayant approximativement la même taille et, pour chacun des ensembles, on teste s'il peut être supprimé. Ce test est effectué en considérant le PL obtenu après suppression, disons $\overline{D}$, et en appliquant l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER. Soit $\overline{y}$ une solution de $\overline{D}$ telle que $b\overline{y} \geq \mathrm{OPT}\left(\overline{D}\right) - \delta$. Si $b\overline{y} \leq by^{(i)} + \delta$, le test est réussi et on pose $D^{(i+1)} := \overline{D}$ et $y^{(i+1)} := \overline{y}$. Choisir δ de manière appropriée.
(Karmarkar et Karp [1982])

∗ 10. Trouver un choix approprié de ϵ sous la forme d'une fonction de $\mathrm{SUM}(I)$, tel que la modification correspondante de l'ALGORITHME DE KARMARKAR-KARP soit un algorithme polynomial qui garantisse de trouver une solution avec au plus $\mathrm{OPT}(I) + O\left(\frac{\mathrm{OPT}(I) \log\log \mathrm{OPT}(I)}{\log \mathrm{OPT}(I)}\right)$ boîtes.
(Johnson [1982])

Références

Littérature générale :

Coffman, E.G., Garey, M.R., Johnson, D.S. [1996] : Approximation algorithms for bin-packing ; a survey. In : Approximation Algorithms for *NP*-Hard Problems (D.S. Hochbaum, ed.), PWS, Boston, 1996

Références citées :

Baker, B.S. [1985] : A new proof for the First-Fit Decreasing bin-packing algorithm. Journal of Algorithms 6 (1985), 49–70

Bansal, N., Correa, J.R., Kenyon, C., Sviridenko, M. [2006] : Bin packing in multiple dimensions : inapproximability results and approximation schemes. Mathematics of Operations Research 31 (2006), 31–49

Caprara, A. [2002] : Packing 2-dimensional bins in harmony. Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (2002), 490–499

Eisemann, K. [1957] : The trim problem. Management Science 3 (1957), 279–284

Fernandez de la Vega, W., Lueker, G.S. [1981] : Bin packing can be solved within $1 + \epsilon$ in linear time. Combinatorica 1 (1981), 349–355

Garey, M.R., Graham, R.L., Johnson, D.S., Yao, A.C. [1976] : Resource constrained scheduling as generalized bin packing. Journal of Combinatorial Theory A 21 (1976), 257–298

Garey, M.R., Johnson, D.S. [1975] : Complexity results for multiprocessor scheduling under resource constraints. SIAM Journal on Computing 4 (1975), 397–411

Garey, M.R., Johnson, D.S. [1979] : Computers and Intractability ; A Guide to the Theory of *NP*-Completeness. Freeman, San Francisco 1979, p. 127

Gilmore, P.C., Gomory, R.E. [1961] : A linear programming approach to the cutting-stock problem. Operations Research 9 (1961), 849–859

Graham, R.L. [1966] : Bounds for certain multiprocessing anomalies. Bell Systems Technical Journal 45 (1966), 1563–1581

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. [1979] : Optimization and approximation in deterministic sequencing and scheduling : a survey. In : Discrete Optimization II ; Annals of Discrete Mathematics 5 (P.L. Hammer, E.L. Johnson, B.H. Korte, eds.), North-Holland, Amsterdam 1979, pp. 287–326

Hochbaum, D.S., Shmoys, D.B. [1987] : Using dual approximation algorithms for scheduling problems : theoretical and practical results. Journal of the ACM 34 (1987), 144–162

Horowitz, E., Sahni, S.K. [1976] : Exact and approximate algorithms for scheduling nonidentical processors. Journal of the ACM 23 (1976), 317–327

Johnson, D.S. [1973] : Near-Optimal Bin Packing Algorithms. Doctoral Thesis, Dept. of Mathematics, MIT, Cambridge, MA, 1973

Johnson, D.S. [1974] : Fast algorithms for bin-packing. Journal of Computer and System Sciences 8 (1974), 272–314

Johnson, D.S. [1982] : The *NP*-completeness column ; an ongoing guide. Journal of Algorithms 3 (1982), 288–300, Section 3

Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L. [1974] : Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on Computing 3 (1974), 299–325

Karmarkar, N., Karp, R.M. [1982] : An efficient approximation scheme for the one-dimensional bin-packing problem. Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (1982), 312–320

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. [1993] : Sequencing and scheduling : algorithms and complexity. In : Handbooks in Operations Research and Management Science ; Vol. 4 (S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin, eds.), Elsevier, Amsterdam 1993

Lenstra, H.W. [1983] : Integer Programming with a fixed number of variables. Mathematics of Operations Research 8 (1983), 538–548

Papadimitriou, C.H. [1994] : Computational Complexity. Addison-Wesley, Reading 1994, pp. 204–205

Plotkin, S.A., Shmoys, D.B., Tardos, É. [1995] : Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research 20 (1995), 257–301

Seiden, S.S. [2002] : On the online bin packing problem. Journal of the ACM 49 (2002), 640–671

Simchi-Levi, D. [1994] : New worst-case results for the bin-packing problem. Naval Research Logistics 41 (1994), 579–585

Van Vliet, A. [1992] : An improved lower bound for on-line bin packing algorithms. Information Processing Letters 43 (1992), 277–284

Yue, M. [1990] : A simple proof of the inequality $FFD(L) \leq \frac{11}{9}\,\mathrm{OPT}(L) + 1, \forall L$ for the FFD bin-packing algorithm. Report No. 90665, Research Institute for Discrete Mathematics, University of Bonn, 1990

Zhang, G. [2005] : A 3-approximation algorithm for two-dimensional bin packing. Operations Research Letters 33 (2005), 121–126

Chapitre 19

Multiflots et chaînes arête-disjointes

Le PROBLÈME DU MULTIFLOT est une généralisation du PROBLÈME DU FLOT MAXIMUM. Étant donné un graphe orienté G avec des capacités u sur les arcs, nous recherchons maintenant un flot de s à t pour plusieurs paires (s,t) (on parle de plusieurs commodités) tel que le flot total passant par un arc n'excède pas sa capacité. Nous représentons les paires (s,t) à l'aide d'un deuxième graphe orienté ; pour des raisons techniques nous ajoutons un arc de t à s lorsque nous recherchons un flot de s à t. Nous avons formellement :

PROBLÈME DU MULTIFLOT ORIENTÉ

Instance Une paire (G,H) de graphes orientés sur le même ensemble de sommets. Des capacités $u : E(G) \to \mathbb{R}_+$ et des demandes $b : E(H) \to \mathbb{R}_+$.

Tâche Trouver une famille $(x^f)_{f\in E(H)}$, telle que x^f soit un flot de s à t de valeur $b(f)$ dans G pour chaque $f = (t,s) \in E(H)$, et tel que

$$\sum_{f\in E(H)} x^f(e) \;\leq\; u(e) \quad \text{pour tout } e \in E(G).$$

Il existe également une version non orientée du problème que nous traiterons plus tard. De nouveau, les arcs de G sont appelés **arcs d'offre** et les arcs de H **arcs de demande**. Si $u \equiv 1$, $b \equiv 1$ et que l'on requiert que x soit à valeurs entières, on obtient le PROBLÈME DES CHEMINS ARC-DISJOINTS. On a parfois également des poids sur les arcs et on recherche alors un multiflot de coût minimum. Mais ici on s'intéresse seulement aux solutions réalisables.

Bien entendu, le problème peut être résolu en temps polynomial à l'aide de la PROGRAMMATION LINÉAIRE (voir théorème 4.18). Cependant les PLs correspon-

dants sont d'assez grande taille, il est donc aussi intéressant d'avoir un algorithme combinatoire pour résoudre le problème approximativement ; voir le paragraphe 19.2. Cet algorithme est fondé sur une formulation par PL. De plus, la dualité fournit une bonne caractérisation de notre problème comme nous le verrons au paragraphe 19.1. Cela donne des conditions de réalisabilité nécessaires (mais en général non suffisantes) pour le PROBLÈME DES CHEMINS ARC-DISJOINTS.

Dans de nombreuses applications, on s'intéresse à des flots à valeurs entières, ou à des chemins et le PROBLÈME DES CHEMINS ARC-DISJOINTS est une formulation adaptée.

Nous avons considéré un cas particulier de ce problème au paragraphe 8.2, où nous avions une condition nécessaire et suffisante pour l'existence de k chemins arc-disjoints (ou sommet-disjoints) de s à t pour deux sommets s et t donnés (théorèmes de Menger 8.9 et 8.10). Nous prouverons que le PROBLÈME DES CHEMINS ARC-DISJOINTS général est *NP*-difficile ainsi que le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES. Néanmoins il existe quelques cas particuliers intéressants qui peuvent être résolus en temps polynomial, comme nous le verrons aux paragraphes 19.3 et 19.4.

19.1 Multiflots

Nous nous concentrons sur le PROBLÈME DU MULTIFLOT ORIENTÉ, mais tous les résultats de cette section sont également valables pour le cas non orienté :

PROBLÈME DU MULTIFLOT NON ORIENTÉ

Instance Une paire (G, H) de graphes non orientés sur le même ensemble de sommets.
Des capacités $u : E(G) \to \mathbb{R}_+$ et des demandes $b : E(H) \to \mathbb{R}_+$.

Tâche Trouver une famille $(x^f)_{f \in E(H)}$, telle que x^f soit un flot de s à t de valeur $b(f)$ dans $(V(G), \{(v,w), (w,v) : (v,w) \in E(G)\})$ pour chaque $f = (t,s) \in E(H)$, et tel que

$$\sum_{f \in E(H)} \left(x^f((v,w)) + x^f((w,v))\right) \leq u(e)$$

pour tout $e = (v,w) \in E(G)$.

Les deux versions du PROBLÈME DU MULTIFLOT peuvent être représentées de manière naturelle sous la forme d'un PL (voir le PL associé au PROBLÈME DU FLOT MAXIMUM du paragraphe 8.1). Elles peuvent donc être résolues en temps polynomial (théorème 4.18). On ne connaît pas aujourd'hui d'algorithmes polynomiaux qui n'utilisent pas la PROGRAMMATION LINÉAIRE sauf pour des cas particuliers.

Nous allons maintenant mentionner une formulation PL différente du PROBLÈME DU MULTIFLOT qui se révèlera utile :

Lemme 19.1. *Soit (G, H, u, b) une instance du* PROBLÈME DU MULTIFLOT (ORIENTÉ *ou* NON ORIENTÉ). *Soit $\mathcal{C}$ l'ensemble des circuits (ou des cycles) de $G + H$ qui contiennent exactement un arc (ou une arête) de demande. Soit M une matrice 0-1 dont les colonnes sont indicées par les éléments de $\mathcal{C}$ et les lignes par les arcs (ou arêtes) de G, de telle sorte que $M_{e,C} = 1$ si et seulement si $e \in C$. De manière similaire, soit N une matrice 0-1 dont les colonnes sont indicées par les éléments de $\mathcal{C}$ et les lignes par les arcs (ou arêtes) de H, de telle sorte que $N_{f,C} = 1$ si et seulement si $f \in C$.*

Alors chaque solution du PROBLÈME DU MULTIFLOT *correspond à au moins un point du polytope*

$$\left\{y \in \mathbb{R}^{\mathcal{C}} : y \geq 0,\ My \leq u,\ Ny = b\right\}, \tag{19.1}$$

et chaque point de ce polytope correspond à une unique solution du PROBLÈME DU MULTIFLOT.

Preuve. Pour simplifier nos notations, nous considérons uniquement le cas orienté ; le cas non orienté est obtenu en remplaçant chaque arête par le sous-graphe représenté à la figure 8.2.

Soit $(x^f)_{f \in E(H)}$ une solution du PROBLÈME DU MULTIFLOT. Pour chaque $f = (t, s) \in E(H)$, le flot de s à t x^f peut être décomposé en un ensemble $\mathcal{P}$ de chemins de s à t et un ensemble $\mathcal{Q}$ de circuits (théorème 8.8) : pour chaque arc de demande f on peut écrire

$$x^f(e) \ = \sum_{P \in \mathcal{P} \cup \mathcal{Q} :\, e \in E(P)} w(P)$$

pour $e \in E(G)$, où $w : \mathcal{P} \cup \mathcal{Q} \to \mathbb{R}_+$. On pose $y_{P+f} := w(P)$ pour $P \in \mathcal{P}$ et $y_C := 0$ pour $f \in C \in \mathcal{C}$ tel que $C - f \notin \mathcal{P}$. Cela fournit évidemment un vecteur $y \geq 0$ tel que $My \leq u$ et $Ny = b$.

Inversement, soit $y \geq 0$ tel que $My \leq u$ et $Ny = b$. En posant

$$x^f(e) \ := \sum_{C \in \mathcal{C} :\, e, f \in E(C)} y_C,$$

on obtient une solution du PROBLÈME DU MULTIFLOT. □

À l'aide de la dualité, on peut maintenant présenter une condition nécessaire et suffisante de l'existence d'une solution au PROBLÈME DU MULTIFLOT. Nous allons également faire le lien avec le PROBLÈME DES CHEMINS ARC-DISJOINTS.

Définition 19.2. *Une instance (G, H) du* PROBLÈME DES CHEMINS ARC-DISJOINTS *(ou du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES*) satisfait le* **critère de distance** *si pour chaque $z : E(G) \to \mathbb{R}_+$*

$$\sum_{f=(t,s) \in E(H)} \mathrm{dist}_{(G,z)}(s, t) \ \leq \sum_{e \in E(G)} z(e). \tag{19.2}$$

Une instance (G, H, u, b) *du* PROBLÈME DU MULTIFLOT *satisfait le* **critère de distance** *si pour chaque* $z : E(G) \to \mathbb{R}_+$

$$\sum_{f=(t,s)\in E(H)} b(f)\,\mathrm{dist}_{(G,z)}(s,t) \;\leq\; \sum_{e\in E(G)} u(e)z(e).$$

Le terme de gauche du critère de distance peut s'interpréter comme une borne inférieure du coût d'une solution (par rapport aux coûts des arcs z), alors que le terme de droite correspond à une borne supérieure du coût maximum possible.

Théorème 19.3. *Le critère de distance constitue une condition nécessaire et suffisante pour l'existence d'une solution au* PROBLÈME DU MULTIFLOT *(dans les cas orientés et non orientés).*

Preuve. Nous considérons de nouveau uniquement le cas orienté. Le cas non orienté se traite par la substitution représentée à la figure 8.2. D'après le lemme 19.1, le PROBLÈME DU MULTIFLOT a une solution si et seulement si le polyèdre $\left\{y \in \mathbb{R}_+^{\mathcal{C}} : My \leq u, Ny = b\right\}$ est non vide. D'après le corollaire 3.25, ce polyèdre est vide si et seulement s'il existe des vecteurs z, w tels que $z \geq 0$, $zM + wN \geq 0$ et $zu + wb < 0$. (M et N sont définis comme précédemment.)

L'inégalité $zM + wN \geq 0$ implique

$$-w_f \;\leq\; \sum_{e\in P} z_e$$

pour chaque arc de demande $f = (t, s)$ et chaque chemin de s à t P de G, donc $-w_f \leq \mathrm{dist}_{(G,z)}(s,t)$. Il existe ainsi des vecteurs z, w tels que $z \geq 0$, $zM + wN \geq 0$ et $zu + wb < 0$ si et seulement s'il existe un vecteur $z \geq 0$ tel que

$$zu - \sum_{f=(t,s)\in E(H)} \mathrm{dist}_{(G,z)}(s,t)\,b(f) \;<\; 0.$$

Cela complète la preuve. □

Au paragraphe 19.2, nous montrerons comment le PL du lemme 19.1 et son dual peuvent être utilisés pour construire un algorithme pour le PROBLÈME DU MULTIFLOT.

Le théorème 19.3 implique que le critère de distance est nécessaire pour l'existence d'une solution du PROBLÈME DES CHEMINS ARC-DISJOINTS, puisqu'il peut être considéré comme un PROBLÈME DE MULTIFLOT avec $b \equiv 1$, $u \equiv 1$ et des contraintes d'intégralité. La condition nécessaire suivante est également importante :

Définition 19.4. *Une instance* (G, H) *du* PROBLÈME DES CHEMINS ARC-DISJOINTS *(ou du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES*) satisfait le* **critère de coupe** *si pour chaque* $X \subseteq V(G)$ *:*

- $|\delta_G^+(X)| \geq |\delta_H^-(X)|$ *dans le cas orienté, ou*
- $|\delta_G(X)| \geq |\delta_H(X)|$ *dans le cas non orienté.*

Corollaire 19.5. *Pour une instance (G, H) du* PROBLÈME DES CHEMINS ARC-DISJOINTS *(ou du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES*), l'implication suivante est vérifiée : (G, H) a une solution $\Rightarrow$ (G, H) satisfait le critère de distance $\Rightarrow$ (G, H) satisfait le critère de coupe.*

Preuve. La première implication est conséquence du théorème 19.3. Pour la seconde implication, observons que le critère de coupe est juste un cas particulier du critère de distance, où on considère des fonctions poids du type

$$z(e) := \begin{cases} 1 & \text{si } e \in \delta^+(X) \text{ (cas orienté) ou } e \in \delta(X) \text{ (cas non orienté)} \\ 0 & \text{sinon} \end{cases}$$

pour $X \subseteq V(G)$. □

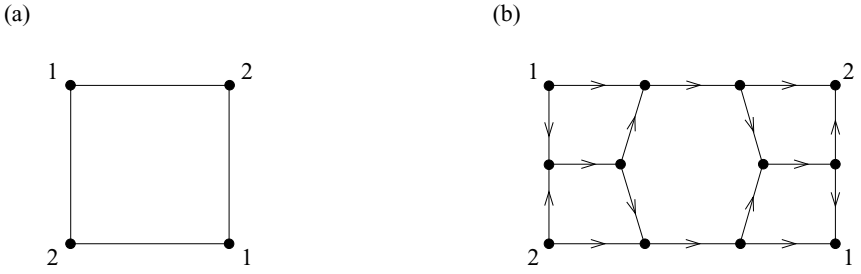


Figure 19.1.

En général aucune de ces implications ne peut être inversée. La figure 19.1 montre des exemples où il n'existe pas de solution (entière), mais il existe une solution fractionnaire, c.-à-d. une solution de la relaxation du PROBLÈME DU MULTIFLOT. Le critère de distance est donc ici vérifié. Dans les figures de ce paragraphe, les arcs de demande sont indiqués par des nombres égaux à leurs extrémités. Dans le cas orienté, on oriente les arcs de demande de telle façon qu'ils soient réalisables. (Un arc (ou une arête) de demande (t, s) est dit **réalisable** si t est connecté à s dans le graphe offre.)

Les deux exemples représentés à la figure 19.2 satisfont le critère de coupe (cela se vérifie facilement), mais aucun ne satisfait le critère de distance : dans le cas non orienté choisir $z(e) = 1$ pour tout $e \in E(G)$, dans l'exemple orienté choisir $z(e) = 1$ pour les arcs en gras et $z(e) = 0$ sinon.

19.2 Algorithmes pour le multiflot

La définition du PROBLÈME DU MULTIFLOT donne directement une formulation du problème sous la forme d'un PL de taille polynomiale. Bien que l'on ait alors un

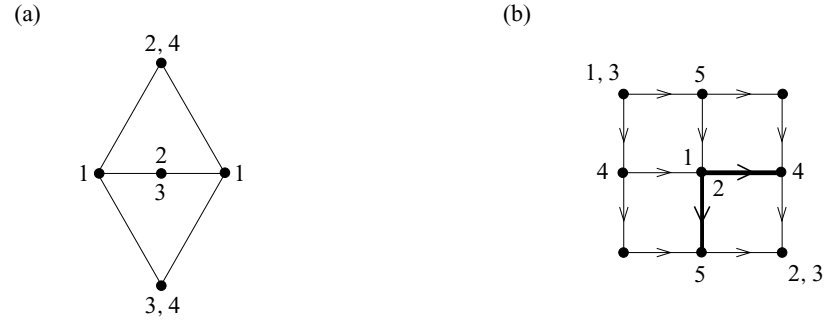


Figure 19.2.

algorithme polynomial pour le résoudre, il ne peut pas être utilisé pour résoudre des instances de grande taille : le nombre de variables est alors trop grand. La description (19.1) donnée au lemme 19.1 paraît même pire puisqu'elle contient un nombre exponentiel de variables. Néanmoins cette description se révèle bien plus utile en pratique. Nous allons expliquer cela maintenant.

Puisque l'on s'intéresse uniquement à une solution réalisable, on considère le PL

$$\max\{0y : y \geq 0,\ My \leq u,\ Ny = b\}$$

et son dual $\min\{zu + wb : z \geq 0,\ zM + wN \geq 0\}$ que l'on peut réécrire sous la forme

$$\min\{zu + wb : z \geq 0,\ \mathrm{dist}_{(G,z)}(s,t) \geq -w(f) \text{ pour tout } f = (t,s) \in E(H)\}.$$

Ce PL dual a seulement $|E(G)| + |E(H)|$ variables, mais un nombre exponentiel de contraintes. Cependant, cela n'a pas d'importance puisque le PROBLÈME DE SÉPARATION peut être résolu à l'aide de $|E(H)|$ calculs de plus courts chemins. Comme on ne doit considérer que les vecteurs z non négatifs, on peut utiliser ici l'ALGORITHME DE DIJKSTRA. Si le PL dual est non borné, alors cela prouve l'irréalisabilité du PL primal. Sinon on peut résoudre le PL dual, mais cela ne fournit généralement pas une solution du primal.

Ford et Fulkerson [1958] ont suggéré d'utiliser la considération précédente pour résoudre le PL primal directement, en association avec l'ALGORITHME DU SIMPLEXE. Puisque la plupart des variables sont nulles à chaque itération de l'ALGORITHME DU SIMPLEXE, on garde seulement trace des variables pour lesquelles la contrainte de positivité $y_C \geq 0$ ne fait pas partie de l'ensemble courant J des lignes actives. Les autres variables ne sont pas stockées explicitement, mais générées lorsque l'on en a besoin (lorsque la contrainte de positivité devient inactive). Le problème qui consiste à décider, à chaque étape, quelle variable doit être générée est équivalent au PROBLÈME DE SÉPARATION pour le PL dual. Il se réduit donc, dans notre cas, à un PROBLÈME DU PLUS COURT CHEMIN. Cette technique de génération de colonnes peut être assez efficace en pratique.

Même avec ces techniques, il y a beaucoup d'instances qui ne peuvent pas être résolues optimalement. Cependant, la méthode précédente fournit également un algorithme d'approximation. Formulons d'abord notre problème comme un problème d'optimisation.

PROBLÈME DU MULTIFLOT MAXIMUM	
Instance	Une paire (G, H) de graphes orientés sur le même ensemble de sommets. Des capacités $u : E(G) \to \mathbb{R}_+$.
Tâche	Trouver une famille $(x^f)_{f \in E(H)}$, telle que x^f soit un flot de s à t dans G pour chaque $f = (t, s) \in E(H)$, que $\sum_{f \in E(H)} x^f(e) \leq u(e)$ pour tout $e \in E(G)$, et que la valeur du flot total $\sum_{f \in E(H)} \text{valeur}(x^f)$ soit maximum.

Il existe d'autres formulations intéressantes. Par exemple, on peut rechercher des flots satisfaisant la plus grande fraction possible de demandes données, ou des flots satisfaisant les demandes, mais violant les capacités aussi peu que possible. De plus on peut considérer des coûts sur les arcs. Nous considérons uniquement le PROBLÈME DU MULTIFLOT MAXIMUM. Les autres problèmes peuvent être traités avec des techniques similaires.

Nous considérons de nouveau le PL

$$\max \left\{ \sum_{P \in \mathcal{P}} y(P) : y \geq 0, \sum_{P \in \mathcal{P} : e \in E(P)} y(P) \leq u(e) \text{ pour tout } e \in E(G) \right\},$$

où $\mathcal{P}$ est la famille des chemins de s à t de G pour tout $(t, s) \in E(H)$, et son dual

$$\min \left\{ zu : z \geq 0, \sum_{e \in E(P)} z(e) \geq 1 \text{ pour tout } P \in \mathcal{P} \right\}.$$

Nous allons décrire un algorithme primal-dual fondé sur ces formulations, qui se révèle être un schéma d'approximation entièrement polynomial. Cet algorithme conserve toujours un vecteur $y \geq 0$ associé au primal qui n'est pas nécessairement une solution réalisable du primal puisque les contraintes de capacité peuvent être violées. Initialement $y = 0$. À la fin, nous multiplierons y par une constante afin que toutes les contraintes soient vérifiées. Pour stocker y de manière efficace, on garde trace de la famille $\mathcal{P}' \subseteq \mathcal{P}$ des chemins P tels que $y(P) > 0$. Contrairement à $\mathcal{P}$, la cardinalité de $\mathcal{P}'$ est bornée polynomialement.

L'algorithme conserve également un vecteur $z \geq 0$ associé au dual. Initialement, $z(e) = \delta$ pour tout $e \in E(G)$, où δ dépend de n et du paramètre d'erreur ϵ. À chaque itération, l'algorithme trouve une contrainte du dual maximalement violée (qui correspond à un plus court chemin de s à t pour $(t, s) \in E(H)$, par rapport aux longueurs des arêtes z) et augmente z et y le long de ce chemin :

SCHÉMA D'APPROXIMATION DU MULTIFLOT

Input Une paire (G, H) de graphes orientés sur le même ensemble de sommets.
Des capacités $u : E(G) \to \mathbb{R}_+ \setminus \{0\}$. Un nombre ϵ tel que $0 < \epsilon \leq \frac{1}{2}$.

Output Des nombres $y : \mathcal{P} \to \mathbb{R}_+$ tels que $\sum_{P \in \mathcal{P} : e \in E(P)} y(P) \leq u(e)$ pour tout $e \in E(G)$.

① Poser $y(P) := 0$ pour tout $P \in \mathcal{P}$.
Poser $\delta := (n(1+\epsilon))^{-\lceil \frac{5}{\epsilon} \rceil}(1+\epsilon)$ et $z(e) := \delta$ pour tout $e \in E(G)$.

② Soit $P \in \mathcal{P}$ tel que $z(E(P))$ soit minimum.
If $z(E(P)) \geq 1$, **then go to** ④.

③ Soit $\gamma := \min_{e \in E(P)} u(e)$.
Poser $y(P) := y(P) + \gamma$.
Poser $z(e) := z(e)\left(1 + \frac{\epsilon\gamma}{u(e)}\right)$ pour tout $e \in E(P)$.
Go to ②.

④ Soit $\xi := \max_{e \in E(G)} \frac{1}{u(e)} \sum_{P \in \mathcal{P} : e \in E(P)} y(P)$.
Poser $y(P) := \frac{y(P)}{\xi}$ pour tout $P \in \mathcal{P}$.

Cet algorithme dû à Young [1995] et Garg et Könemann [1998] est fondé sur les travaux précédents de Shahrokhi et Matula [1990], Shmoys [1996] et d'autres.

Théorème 19.6. (Garg et Könemann [1998]) *Le* SCHÉMA D'APPROXIMATION DU MULTIFLOT *produit une solution réalisable dont la valeur du flot total est au moins égale à* $\frac{1}{1+\epsilon}\,\mathrm{OPT}(G, H, u)$. *Sa complexité est* $O\left(\frac{1}{\epsilon^2} km(m + n\log n)\log n\right)$, *où* $k = |E(H)|$, $n = |V(G)|$ *et* $m = |E(G)|$. *Il s'agit donc d'un schéma d'approximation entièrement polynomial.*

Preuve. À chaque itération, la valeur $z(e)$ augmente d'un facteur $1 + \epsilon$ pour au moins un arc e (l'arc seuil). Puisqu'un arc e tel que $z(e) \geq 1$ n'est plus jamais utilisé dans un chemin, le nombre total d'itérations t est inférieur ou égal à $m\lceil \log_{1+\epsilon}(\frac{1}{\delta}) \rceil$. À chaque itération, on doit résoudre k instances du PROBLÈME DU PLUS COURT CHEMIN avec des poids non négatifs pour déterminer P. En utilisant l'ALGORITHME DE DIJKSTRA (théorème 7.4), on obtient un temps de calcul total en $O(tk(m + n\log n)) = O\left(km(m + n\log n)\log_{1+\epsilon}(\frac{1}{\delta})\right)$. On obtient finalement le temps de calcul annoncé en observant que, pour $0 < \epsilon \leq 1$,

$$\log_{1+\epsilon}\left(\frac{1}{\delta}\right) \;=\; \frac{\log(\frac{1}{\delta})}{\log(1+\epsilon)} \;\leq\; \frac{\lceil \frac{5}{\epsilon} \rceil \log(2n)}{\frac{\epsilon}{2}} \;=\; O\left(\frac{\log n}{\epsilon^2}\right);$$

nous avons utilisé ici que $\log(1+\epsilon) \geq \frac{\epsilon}{2}$ pour $0 < \epsilon \leq 1$.

Nous devons aussi vérifier que le nombre maximum de bits nécessaires pour stocker un nombre apparaissant au cours du calcul est borné par un polynôme par

rapport à $\log n + \text{taille}(u) + \text{taille}(\epsilon) + \frac{1}{\epsilon}$. Cela est évident pour les variables y. Le nombre δ peut être stocké à l'aide de $O(\frac{1}{\epsilon}\,\text{taille}(n(1+\epsilon)) + \text{taille}(\epsilon)) = O(\frac{1}{\epsilon}(\log n + \text{taille}(\epsilon)))$ bits. Pour traiter le cas des variables z, on suppose que u est à valeurs entières ; sinon on multiplie au départ toutes les capacités par le produit des dénominateurs (voir proposition 4.1). Le dénominateur des variables z est alors borné à tout instant par le produit de toutes les capacités et le dénominateur de δ. Puisque le numérateur est au plus égal à deux fois le dénominateur, on a montré que la taille de tous les nombres est effectivement polynomiale par rapport à la taille de l'input et à $\frac{1}{\epsilon}$.

La réalisabilité de la solution est garantie par l'étape ④.

Remarquons qu'à chaque fois que l'on ajoute γ unités de flot sur l'arête e, on augmente le poids $z(e)$ d'un facteur $\left(1 + \frac{\epsilon\gamma}{u(e)}\right)$. Cette valeur est au moins égale à $(1+\epsilon)^{\frac{\gamma}{u(e)}}$, car l'inégalité $1 + \epsilon a \geq (1+\epsilon)^a$ est vérifiée pour $0 \leq a \leq 1$ (les deux termes de cette inégalité sont égaux pour $a \in \{0,1\}$ et le terme de gauche est linéaire par rapport à a tandis que le terme de droite est convexe). Puisque e n'est plus utilisé une fois que $z(e) \geq 1$, on ne peut ajouter plus de $u(e)(1 + \log_{1+\epsilon}(\frac{1}{\delta}))$ unités de flot sur l'arête e. Ainsi

$$\xi \;\leq\; 1 + \log_{1+\epsilon}\left(\frac{1}{\delta}\right) \;=\; \log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right). \tag{19.3}$$

Notons $z^{(i)}$ le vecteur z après l'itération i et notons P_i et γ_i le chemin P et le nombre γ à l'itération i. On a $z^{(i)}u = z^{(i-1)}u + \epsilon\gamma_i \sum_{e\in E(P_i)} z^{(i-1)}(e)$, donc $(z^{(i)} - z^{(0)})u = \epsilon \sum_{j=1}^{i} \gamma_j \alpha(z^{(j-1)})$, où $\alpha(z) := \min_{P\in\mathcal{P}} z(E(P))$. Définissons $\beta := \min\left\{zu : z \in \mathbb{R}_+^{E(G)},\ \alpha(z) \geq 1\right\}$. Alors $(z^{(i)} - z^{(0)})u \geq \beta\alpha(z^{(i)} - z^{(0)})$ et ainsi $(\alpha(z^{(i)}) - \delta n)\beta \leq \alpha(z^{(i)} - z^{(0)})\beta \leq (z^{(i)} - z^{(0)})u$. On obtient

$$\alpha(z^{(i)}) \;\leq\; \delta n + \frac{\epsilon}{\beta}\sum_{j=1}^{i} \gamma_j \alpha(z^{(j-1)}). \tag{19.4}$$

Nous prouvons maintenant

$$\delta n + \frac{\epsilon}{\beta}\sum_{j=1}^{i} \gamma_j \alpha(z^{(j-1)}) \;\leq\; \delta n e^{\left(\frac{\epsilon}{\beta}\sum_{j=1}^{i}\gamma_j\right)} \tag{19.5}$$

par induction sur i (on note ici e la base du logarithme naturel). Le cas $i = 0$ est évident. Pour $i > 0$ on a

$$\begin{aligned}
\delta n + \frac{\epsilon}{\beta}\sum_{j=1}^{i} \gamma_j \alpha(z^{(j-1)}) \;&=\; \delta n + \frac{\epsilon}{\beta}\sum_{j=1}^{i-1} \gamma_j \alpha(z^{(j-1)}) + \frac{\epsilon}{\beta}\gamma_i \alpha(z^{(i-1)}) \\
&\leq\; \left(1 + \frac{\epsilon}{\beta}\gamma_i\right) \delta n e^{\left(\frac{\epsilon}{\beta}\sum_{j=1}^{i-1}\gamma_j\right)},
\end{aligned}$$

en utilisant (19.4) et l'hypothèse d'induction. En utilisant l'inégalité $1 + x < e^x$ pour tout $x > 0$ on termine la preuve de (19.5).

En particulier, on conclut de (19.4), (19.5) et du critère d'arrêt que

$$1 \;\leq\; \alpha(z^{(t)}) \;\leq\; \delta n e^{\left(\frac{\epsilon}{\beta}\sum_{j=1}^{t}\gamma_j\right)},$$

ainsi $\sum_{j=1}^{t}\gamma_j \;\geq\; \frac{\beta}{\epsilon}\ln\left(\frac{1}{\delta n}\right)$. Observons alors que la valeur du flot total que l'algorithme calcule est égale à $\sum_{P\in\mathcal{P}} y(P) = \frac{1}{\xi}\sum_{j=1}^{t}\gamma_j$. D'après ce qui précéde et (19.3), cela est supérieur ou égal à

$$\begin{aligned}
\frac{\beta\ln\left(\frac{1}{\delta n}\right)}{\epsilon\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)} &= \frac{\beta\ln(1+\epsilon)}{\epsilon}\cdot\frac{\ln\left(\frac{1}{\delta n}\right)}{\ln\left(\frac{1+\epsilon}{\delta}\right)} \\
&= \frac{\beta\ln(1+\epsilon)}{\epsilon}\cdot\frac{\left(\left\lceil\frac{5}{\epsilon}\right\rceil-1\right)\ln(n(1+\epsilon))}{\left\lceil\frac{5}{\epsilon}\right\rceil\ln(n(1+\epsilon))} \\
&\geq \frac{\beta\left(1-\frac{\epsilon}{5}\right)\ln(1+\epsilon)}{\epsilon}
\end{aligned}$$

d'après le choix de δ. Observons alors que β est la valeur optimale du PL dual et ainsi, par le théorème de dualité 3.20, la valeur optimale d'une solution du primal. De plus, $\ln(1+\epsilon) \geq \epsilon - \frac{\epsilon^2}{2}$ (cette inégalité est évidente pour $\epsilon = 0$ et la dérivée du terme de gauche est supérieure à celle du terme de droite pour tout $\epsilon > 0$). Ainsi

$$\frac{\left(1-\frac{\epsilon}{5}\right)\ln(1+\epsilon)}{\epsilon} \;\geq\; \left(1-\frac{\epsilon}{5}\right)\left(1-\frac{\epsilon}{2}\right) \;=\; \frac{1+\frac{3}{10}\epsilon-\frac{6}{10}\epsilon^2+\frac{1}{10}\epsilon^3}{1+\epsilon} \;\geq\; \frac{1}{1+\epsilon}$$

pour $\epsilon \leq \frac{1}{2}$. On en conclut que l'algorithme trouve une solution dont la valeur du flot total est au moins égale à $\frac{1}{1+\epsilon}\,\mathrm{OPT}(G,H,u)$. □

Un algorithme différent, mais qui donne le même résultat (avec une analyse plus difficile) a été publié auparavant par Grigoriadis et Khachiyan [1996]. Fleischer [2000] a amélioré le temps de calcul de l'algorithme précédent d'un facteur k. Elle a observé qu'il suffit de calculer une solution approchée du PROBLÈME DU PLUS COURT CHEMIN à l'étape ② et a utilisé ce fait pour montrer qu'il n'est pas nécessaire de calculer un plus court chemin pour chaque paire $(t,s) \in E(H)$ à chaque itération. Voir également Karakostas [2002], Vygen [2004], Bienstock et Iyengar [2006] et Chudak et Eleutério [2005].

19.3 Problème des chemins arc-disjoints

Nous commençons par remarquer que le problème est déjà *NP*-difficile dans une version assez restrictive :

Théorème 19.7. (Even, Itai et Shamir [1976]) *Le* PROBLÈME DES CHEMINS ARC-DISJOINTS *est NP-difficile même si G est sans circuits et si H est seulement constitué de deux ensembles d'arcs parallèles.*

Preuve. On transforme polynomialement le PROBLÈME DE LA SATISFAISABILITÉ en notre problème. Étant donné une famille $\mathcal{Z} = \{Z_1, \ldots, Z_m\}$ de clauses sur $X = \{x_1, \ldots, x_n\}$, on construit une instance (G, H) du PROBLÈME DES CHEMINS ARC-DISJOINTS telle que G soit sans circuit, que H soit seulement constitué de deux ensembles d'arcs parallèles et que (G, H) ait une solution si et seulement si $\mathcal{Z}$ est satisfaisable.

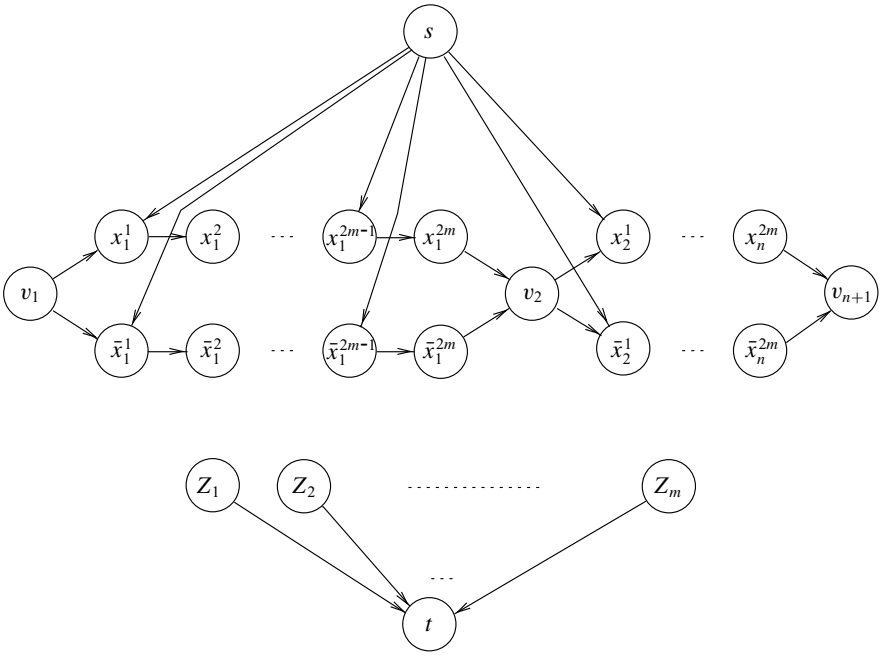


Figure 19.3.

G contient $2m$ sommets $\lambda^1, \ldots, \lambda^{2m}$ associés à chaque littéral et des sommets supplémentaires s et t, $v_1, \ldots, v_{n+1}$ et $Z_1, \ldots, Z_m$. Il y a des arcs (v_i, x_i^1), $(v_i, \overline{x_i}^1)$, (x_i^{2m}, v_{i+1}), $(\overline{x_i}^{2m}, v_{i+1})$, (x_i^j, x_i^{j+1}) et $(\overline{x_i}^j, \overline{x_i}^{j+1})$ pour $i = 1, \ldots, n$ et $j = 1, \ldots, 2m-1$. Ensuite, il y a des arcs (s, x_i^{2j-1}) et $(s, \overline{x_i}^{2j-1})$ pour $i = 1, \ldots, n$ et $j = 1, \ldots, m$. De plus, il y a des arcs (Z_j, t) et (λ^{2j}, Z_j) pour $j = 1, \ldots, m$ et tous les littéraux λ de la clause Z_j. (Voir figure 19.3 pour une illustration).

Soit H l'ensemble constitué d'un arc (v_{n+1}, v_1) et de m arcs parallèles (t, s).

Nous montrons que toute solution de (G, H) correspond à un assignement satisfaisant toutes les clauses (et vice versa). Le chemin de v_1 àv_{n+1} doit passer à travers soit tous les x_i^j (ce qui signifie que x_i est *faux*) soit tous les $\bar{x}_i^j$ (ce qui signifie que x_i est *vrai*) pour chaque i. Un chemin de s à t doit passer à travers chaque Z_j. Cela est possible si et seulement si l'assignement défini ci-dessus satisfait Z_j. □

Fortune, Hopcroft et Wyllie [1980] ont montré que le PROBLÈME DES CHEMINS ARC-DISJOINTS peut être résolu en temps polynomial si G est sans circuits et si $|E(H)| = k$ pour un certain entier k fixé. Si G n'est pas sans circuits, ils ont prouvé que le problème est déjà *NP*-difficile lorsque $|E(H)| = 2$. D'autre part, nous avons :

Théorème 19.8. (Nash-Williams [1969]) *Soit (G, H) une instance du* PROBLÈME DES CHEMINS ARC-DISJOINTS *, telle que $G + H$ soit eulérien et que H soit seulement constitué de deux ensembles d'arcs parallèles. Alors (G, H) a une solution si et seulement si le critère de coupe est vérifié.*

Preuve. On trouve d'abord un ensemble de chemins qui correspondent au premier ensemble d'arcs parallèles de H à l'aide du théorème de Menger 8.9. Après avoir supprimé ces chemins (et les arcs de demande correspondants), l'instance restante satisfait les conditions de la proposition 8.12 et a donc une solution. □

Si $G + H$ est eulérien et que $|E(H)| = 3$, il existe également un algorithme polynomial (Ibaraki et Poljak [1991]). On a d'autre part le résultat plus négatif suivant :

Théorème 19.9. (Vygen [1995]) *Le* PROBLÈME DES CHEMINS ARC-DISJOINTS *est NP-difficile même si G est sans circuits, $G+H$ eulérien et H seulement constitué uniquement de trois ensembles d'arcs parallèles.*

Preuve. On réduit le problème du théorème 19.7 à celui-ci. Soit donc (G, H) une instance du PROBLÈME DES CHEMINS ARC-DISJOINTS, telle que G soit sans circuits et H constitué seulement de deux ensembles d'arcs parallèles.

Pour chaque $v \in V(G)$, on définit

$$\begin{aligned} \alpha(v) &:= \max(0, |\delta^+_{G+H}(v)| - |\delta^-_{G+H}(v)|) \text{ et} \\ \beta(v) &:= \max(0, |\delta^-_{G+H}(v)| - |\delta^+_{G+H}(v)|). \end{aligned}$$

On a

$$\sum_{v \in V(G)} (\alpha(v) - \beta(v)) = \sum_{v \in V(G)} \left(|\delta^+_{G+H}(v)| - |\delta^-_{G+H}(v)|\right) = 0,$$

ce qui implique

$$\sum_{v \in V(G)} \alpha(v) = \sum_{v \in V(G)} \beta(v) =: q.$$

On construit maintenant une instance (G', H') du PROBLÈME DES CHEMINS ARC-DISJOINTS. G' est obtenu de G en ajoutant deux sommets s et t ainsi que $\alpha(v)$ arcs parallèles (s, v) et $\beta(v)$ arcs parallèles (v, t) pour chaque sommet v. H' est constitué de tous les arcs de H et de q arcs parallèles (t, s).

Cette construction peut évidemment être effectuée en temps polynomial. En particulier, le nombre d'arcs dans $G+H$ est au plus multiplié par quatre. De plus, G' est sans circuits, $G' + H'$ est eulérien et H' est seulement constitué de trois ensembles

d'arcs parallèles. Il reste donc à montrer que (G, H) a une solution si et seulement si (G', H') a une solution.

Chaque solution de (G', H') fournit une solution de (G, H) en omettant simplement les chemins de s à t. Considérons donc une solution $\mathcal{P}$ de (G, H). Soit G'' le graphe obtenu de G' en supprimant tous les arcs utilisés par $\mathcal{P}$. Soit H'' le sous-graphe de H' constitué seulement des q arcs allant de t à s. (G'', H'') vérifie les conditions de la proposition 8.12 et a donc une solution. En combinant $\mathcal{P}$ avec une solution de (G'', H''), on obtient une solution de (G', H'). □

Puisqu'une solution d'une instance du PROBLÈME DES CHEMINS ARC-DISJOINTS est constituée de circuits arc-disjoints, il est naturel de se demander combien de circuits arc-disjoints un graphe orienté contient. Nous avons une bonne caractérisation au moins pour les graphes orientés planaires. Plus précisément, on considère le graphe dual planaire et on recherche le nombre maximum de coupes orientées arc-disjointes. Nous avons le théorème min-max bien connu suivant (que nous prouvons de manière très similaire au théorème 6.13) :

Théorème 19.10. (Lucchesi et Younger [1978]) *Soit G un graphe orienté connexe mais pas fortement connexe. Alors le nombre maximum de coupes orientées arc-disjointes dans G est égal à la cardinalité minimum d'un ensemble d'arcs qui contient au moins un élément de chaque coupe orientée.*

Preuve. Soit A la matrice dont les colonnes sont indicées par les arcs et les lignes par les vecteurs d'incidence des coupes orientées. Considérons le PL

$$\min\{\mathbb{1}x : Ax \geq \mathbb{1},\ x \geq 0\},$$

et son dual

$$\max\{\mathbb{1}y : yA \leq \mathbb{1},\ y \geq 0\}.$$

Nous devons alors prouver que le PL primal et son dual ont des solutions optimales entières. D'après le corollaire 5.15, il suffit de montrer que le système $Ax \geq \mathbb{1},\ x \geq 0$ est TDI. On utilise le lemme 5.23.

Soit $c : E(G) \to \mathbb{Z}_+$ et soit y une solution optimale de $\max\{\mathbb{1}y : yA \leq c, y \geq 0\}$ pour laquelle

$$\sum_X y_{\delta^+(X)}|X|^2 \tag{19.6}$$

est aussi grand que possible, où la somme est prise sur toutes les lignes de A. Nous affirmons que le système d'ensemble $(V(G), \mathcal{F})$ tel que $\mathcal{F} := \{X : y_{\delta^+(X)} > 0\}$ est sans croisements. Pour voir cela , considérons $X, Y \in \mathcal{F}$ tels que $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$, $Y \setminus X \neq \emptyset$ et $X \cup Y \neq V(G)$. Alors $\delta^+(X \cap Y)$ et $\delta^+(X \cup Y)$ sont également des coupes orientées (d'après le lemme 2.1(b)). Soit $\epsilon := \min\{y_{\delta^+(X)}, y_{\delta^+(Y)}\}$. On pose $y'_{\delta^+(X)} := y_{\delta^+(X)} - \epsilon$, $y'_{\delta^+(Y)} := y_{\delta^+(Y)} - \epsilon$, $y'_{\delta^+(X\cap Y)} := y_{\delta^+(X\cap Y)} + \epsilon$, $y'_{\delta^+(X\cup Y)} := y_{\delta^+(X\cup Y)} + \epsilon$, et $y'(S) := y(S)$ pour toutes les autres coupes orientées S. Puisque y' est une solution réalisable du dual, elle est aussi optimale et contredit le choix de y, car (19.6) est plus grand pour y'.

Soit alors A' la sous-matrice de A constituée des lignes indicées par les éléments de $\mathcal{F}$. A' est la matrice d'incidence des coupes d'une famille sans croisements. Donc, d'après le théorème 5.28, A' est totalement unimodulaire, comme requis. □

Pour une preuve combinatoire, voir Lovász [1976]. Frank [1981] a donné une preuve algorithmique.

Remarquons que les ensembles d'arcs qui rencontrent toutes les coupes orientées sont précisément les ensembles d'arcs dont la contraction rend le graphe fortement connexe. Dans le graphe planaire dual, ces ensembles correspondent aux ensembles d'arcs qui rencontrent tous les circuits. De tels ensembles sont appelés des **arc-transversaux des circuits** (en anglais *feedback edge sets*). La cardinalité minimum d'un arc-transversal est le **nombre arc-transversal des circuits** (en anglais *feedback number*) du graphe. Le problème de la détermination de ce nombre est généralement *NP*-difficile (Karp [1972]), mais on peut le résoudre polynomialement pour les graphes planaires.

Corollaire 19.11. *Dans un graphe planaire orienté, le nombre maximum de circuits arc-disjoints est égal au nombre minimum d'arcs rencontrant tous les circuits.*

Preuve. Soit G un graphe orienté qui, sans perte de généralité, est connexe et ne contient pas de sommet d'articulation. Considérer le dual planaire de G et le corollaire 2.44 et appliquer le théorème de Lucchesi-Younger 19.10. □

Un algorithme polynomial pour déterminer le nombre arc-transversal des circuits pour les graphes planaires peut être obtenu en associant l'algorithme de planarité (théorème 2.40), l'ALGORITHME DE GRÖTSCHEL-LOVÁSZ-SCHRIJVER (théorème 4.21) et un algorithme pour le PROBLÈME DU FLOT MAXIMUM pour résoudre le PROBLÈME DE SÉPARATION (exercice 4). Une application du PROBLÈME DES CHEMINS ARC-DISJOINTS est la suivante :

Corollaire 19.12. *Soit (G, H) une instance du* PROBLÈME DES CHEMINS ARC-DISJOINTS *, telle que G soit sans circuits et tel que $G + H$ soit planaire. Alors (G, H) a une solution si et seulement si la suppression de $|E(H)| - 1$ arcs quelconques de $G + H$ ne rend pas $G + H$ sans circuits.* □

En particulier, le critère de distance est nécessaire et suffisant dans ce cas et le problème peut être résolu en temps polynomial.

19.4 Problème des chaînes arête-disjointes

Le lemme suivant établit un lien entre les problèmes orientés et non orientés.

Lemme 19.13. *Soit (G, H) une instance du* PROBLÈME DES CHEMINS ARC-DISJOINTS *, telle que G soit sans circuits et $G+H$ eulérien. Considérons l'instance (G', H') du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES *qui est obtenue en supprimant les orientations. Alors chaque solution de (G', H') est aussi une solution de (G, H) et vice versa.*

Preuve. Il est évident que chaque solution de (G, H) est aussi une solution de (G', H'). On prouve l'autre sens par induction sur $|E(G)|$. Si G n'a pas d'arcs, on a terminé.

Soit maintenant $\mathcal{P}$ une solution de (G', H'). Puisque G est sans circuits, G doit contenir un sommet v pour lequel $\delta^-_G(v) = \emptyset$. Puisque $G + H$ est eulérien, on a $|\delta^-_H(v)| = |\delta^+_G(v)| + |\delta^+_H(v)|$.

Pour chaque arc de demande incident à v il doit y avoir une chaîne dans $\mathcal{P}$ commençant en v. Donc $|\delta^+_G(v)| \geq |\delta^-_H(v)| + |\delta^+_H(v)|$. Cela implique $|\delta^+_H(v)| = 0$ et $|\delta^+_G(v)| = |\delta^-_H(v)|$. Ainsi chaque arc incident à v est utilisé par $\mathcal{P}$ avec l'orientation correcte.

Soit alors G_1 le graphe obtenu de G en supprimant les arcs incidents à v. Soit H_1 obtenu de H en remplaçant chaque arc $f = (t, v)$ incident à v par (t, w), où w est le premier sommet intérieur de la chaîne de $\mathcal{P}$ qui réalise f.

Évidemment G_1 est sans circuit et $G_1 + H_1$ est eulérien. Soit $\mathcal{P}_1$ obtenu de $\mathcal{P}$ en supprimant tous les arcs incidents à v. $\mathcal{P}_1$ est une solution de (G'_1, H'_1), le problème non orienté correspondant à (G_1, H_1).

Par l'hypothèse d'induction, $\mathcal{P}_1$ est une solution de (G_1, H_1). Donc, en ajoutant les arêtes initiales, on obtient que $\mathcal{P}$ est une solution de (G, H). □

On conclut :

Théorème 19.14. (Vygen [1995]) *Le* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES *est NP-difficile même si $G + H$ est eulérien et H seulement constitué de trois ensembles d'arêtes parallèles.*

Preuve. On réduit le problème du théorème 19.9 au cas non orienté en appliquant le lemme 19.13. □

Le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES est également *NP*-difficile dans le cas particulier où $G + H$ est planaire (Middendorf et Pfeiffer [1993]). Cependant, si $G + H$ est planaire et eulérien, alors on peut résoudre le problème polynomialement. On a le résultat suivant :

Théorème 19.15. (Seymour [1981]) *Soit (G, H) une instance du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES, *telle que $G+H$ soit planaire et eulérien. Alors (G, H) a une solution si et seulement si le critère de coupe est vérifié.*

Preuve. On doit seulement prouver la condition suffisante du critère de coupe. On peut supposer que $G + H$ est connexe. Soit D le dual planaire $G + H$. Soit $F \subseteq E(D)$ l'ensemble des arêtes du dual correspondant aux arêtes de demande. Alors le critère de coupe et le théorème 2.43 impliquent que $|F \cap E(C)| \leq |E(C) \setminus F|$ pour chaque cycle C de D. Donc, d'après la proposition 12.7, F est un T-joint minimum, où $T := \{x \in V(D) : |F \cap \delta(x)| \text{ est impaire}\}$.

Puisque $G + H$ est eulérien, D est biparti d'après le corollaire 2.45. Donc, d'après le théorème 12.15, il existe $|F|$ T-coupes arête-disjointes $C_1, \ldots, C_{|F|}$. Puisque d'après la proposition 12.14 chaque T-coupe intersecte F, chacune des T-coupes $C_1, \ldots C_{|F|}$ doit contenir exactement une arête de F.

Si on considère de nouveau $G + H$, les duals de $C_1, \ldots, C_{|F|}$ correspondent à des cycles arête-disjoints, chacun contenant exactement une arête de demande. Mais cela signifie que l'on a une solution du PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES. □

Ce théorème implique aussi un algorithme polynomial (exercice 8). En fait, Matsumoto, Nishizeki et Saito [1986] ont prouvé que le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES avec $G + H$ planaire et eulérien peut être résolu en temps $O\big(n^{\frac{5}{2}} \log n\big)$.

D'autre part, Robertson et Seymour ont trouvé un algorithme polynomial dans le cas où le nombre d'arêtes de demande est fixé :

Théorème 19.16. (Robertson et Seymour [1995]) *Pour k fixé, il existe un algorithme polynomial pour le* PROBLÈME DES CHAÎNES SOMMET-DISJOINTES *ou* ARÊTE-DISJOINTES *restreint aux instances telles que $|E(H)| \leq k$.*

Remarquons que le PROBLÈME DES CHAÎNES SOMMET-DISJOINTES est également *NP*-difficile ; voir l'exercice 11. Le théorème 19.16 fait partie de l'importante série d'articles de Robertson et Seymour sur les mineurs des graphes qui sort du domaine couvert par ce livre. Le théorème a été prouvé dans le cas où les chaînes sont sommet-disjointes. Robertson et Seymour ont prouvé que soit il existe un sommet inutile (qui peut être supprimé sans que cela affecte l'existence d'une solution), soit le graphe a une décomposition arborescente de petite largeur (dans ce cas il existe un algorithme polynomial simple ; voir exercice 10). On peut facilement voir que le cas où les chaînes sont arête-disjointes s'obtient comme conséquence de ce résultat ; voir l'exercice 11. Bien que le temps de calcul soit $O(n^2 m)$, la constante dépendant de k augmente extrêmement rapidement et on ne peut plus utiliser l'algorithme en pratique déjà avec $k = 3$.

Le reste de ce paragraphe est consacré aux preuves de deux résultats supplémentaires importants. Le premier est le théorème bien connu d'Okamura-Seymour :

Théorème 19.17. (Okamura et Seymour [1981]) *Soit (G, H) une instance du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES, *telle que $G + H$ soit eulérien, que G soit planaire, et que tous les nœuds terminaux soient situés sur la face extérieure. Alors (G, H) a une solution si et seulement si le critère de coupe est vérifié.*

Preuve. On montre que le critère de coupe est une condition suffisante par induction sur $|V(G)| + |E(G)|$. Si $|V(G)| \leq 2$, cela est évident.

On peut supposer que G est 2-connexe, car sinon on peut appliquer l'hypothèse d'induction aux blocs de G (en supprimant les arêtes de demande qui joignent des blocs différents en des sommets d'articulation). On fixe une représentation plane de G. D'après la proposition 2.31 la face extérieure est bordée par un cycle C.

S'il n'existe pas d'ensemble $X \subset V(G)$ tel que $\emptyset \neq X \cap V(C) \neq V(C)$ et $|\delta_G(X)| = |\delta_H(X)|$, alors, pour toute arête $e \in E(C)$, l'instance $(G - e, H + e)$ satisfait le critère de coupe. Cela est vrai, car $|\delta_G(X)| - |\delta_H(X)|$ est pair pour tout $X \subseteq V(G)$ (car $G + H$ est eulérien). Par l'hypothèse d'induction, $(G - e, H + e)$ a une solution qui implique de manière immédiate une solution pour (G, H).

Supposons donc qu'il existe un ensemble $X \subset V(G)$ tel que $\emptyset \neq X \cap V(C) \neq V(C)$ et $|\delta_G(X)| = |\delta_H(X)|$. Choisissons X tel que le nombre total de composantes connexes de $G[X]$ et $G[V(G) \setminus X]$ soit minimum. Il est alors facile de voir que $G[X]$ et $G[V(G) \setminus X]$ sont tous deux connexes : supposons que ce ne soit pas le cas, et que $G[X]$, par exemple, ne soit pas connexe (l'autre cas est symétrique). Alors $|\delta_G(X_i)| = |\delta_H(X_i)|$ pour chaque composante connexe X_i de $G[X]$ et en remplaçant X par X_i (pour un certain i tel que $X_i \cap V(C) \neq \emptyset$) on réduit le nombre de composantes connexes de $G[X]$ sans augmenter le nombre de composantes connexes de $G[V(G) \setminus X]$. Cela contredit le choix de X.

Puisque G est planaire, un ensemble $X \subset V(G)$, tel que $\emptyset \neq X \cap V(C) \neq V(C)$ et tel que $G[X]$ et $G[V(G) \setminus X]$ soient tous deux connexes, a la propriété que $C[X]$ est une chaîne.

Considérons donc $\emptyset \neq X \subseteq V(G)$ tel que $|\delta_G(X)| = |\delta_H(X)|$ et tel que $C[X]$ soit une chaîne de longueur minimum. Numérotons les sommets de C de manière cyclique $v_1, \ldots, v_l$ et de telle façon que $V(C) \cap X = \{v_1, \ldots, v_j\}$. Soit $e := (v_l, v_1)$.

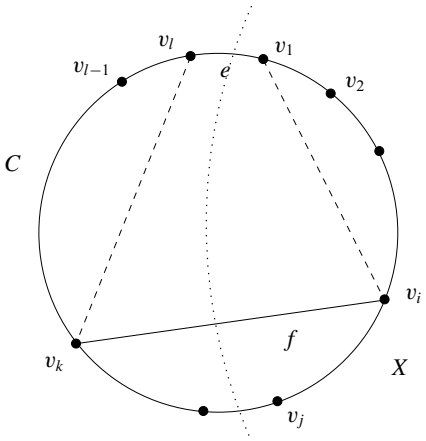


Figure 19.4.

Choisissons $f = (v_i, v_k) \in E(H)$ tel que $1 \leq i \leq j < k \leq l$ (c.-à-d. $v_i \in X$, $v_k \notin X$) et que k soit aussi grand que possible (voir figure 19.4). Considérons alors $G' := G - e$ et $H' := (V(H), (E(H) \setminus \{f\}) \cup \{(v_i, v_1), (v_l, v_k)\})$. (Les cas où $i = 1$ ou $k = l$ ne sont pas exclus, mais nous n'ajoutons pas les boucles correspondantes.)

Nous affirmons que (G', H') satisfait le critère de coupe. Alors, par induction, (G', H') a une solution et elle peut être facilement transformée en une solution de (G, H).

Supposons alors que (G', H') ne satisfait pas le critère de coupe, c.-à-d. que $|\delta_{G'}(Y)| < |\delta_{H'}(Y)|$ pour un certain $Y \subseteq V(G)$. Comme précédemment, on peut supposer que $G[Y]$ et $G[V(G) \setminus Y]$ sont tous deux connexes. En interchangeant si

besoin Y et $V(G) \setminus Y$, on peut aussi supposer que $v_i \notin Y$. Puisque $Y \cap V(C)$ est une chaîne et que $|\delta_{H'}(Y)| - |\delta_{G'}(Y)| > |\delta_H(Y)| - |\delta_G(Y)|$, il y a trois cas :

(a) $v_1 \in Y, v_i, v_k, v_l \notin Y$.

(b) $v_1, v_l \in Y, v_i, v_k \notin Y$.

(c) $v_l \in Y, v_1, v_i, v_k \notin Y$.

Dans chaque cas on a $Y \cap V(C) \subseteq \{v_{k+1}, \ldots, v_{i-1}\}$ donc, d'après le choix de f, on a $E_H(X, Y) = \emptyset$. De plus $|\delta_G(Y)| = |\delta_H(Y)|$. En appliquant deux fois le lemme 2.1(c), on a

$$\begin{aligned} |\delta_H(X)| + |\delta_H(Y)| &= |\delta_G(X)| + |\delta_G(Y)| \\ &= |\delta_G(X \cap Y)| + |\delta_G(X \cup Y)| + 2|E_G(X, Y)| \\ &\geq |\delta_H(X \cap Y)| + |\delta_H(X \cup Y)| + 2|E_G(X, Y)| \\ &= |\delta_H(X)| + |\delta_H(Y)| - 2|E_H(X, Y)| + 2|E_G(X, Y)| \\ &= |\delta_H(X)| + |\delta_H(Y)| + 2|E_G(X, Y)| \\ &\geq |\delta_H(X)| + |\delta_H(Y)| \, . \end{aligned}$$

On doit donc avoir égalité partout. Cela implique $|\delta_G(X \cap Y)| = |\delta_H(X \cap Y)|$ et $E_G(X, Y) = \emptyset$.

Le cas (c) est donc impossible (car ici $e \in E_G(X, Y)$) ; c.-à-d. $v_1 \in Y$. Ainsi $X \cap Y$ est non vide et $C[X \cap Y]$ est une chaîne plus courte que $C[X]$, ce qui contredit le choix de X. □

Cette preuve fournit un algorithme polynomial (exercice 12) pour le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES dans ce cas particulier. Il peut être implémenté en temps $O(n^2)$ (Becker et Mehlhorn [1986]) et même en temps linéaire (Wagner et Weihe [1995]).

Avant de présenter le deuxième résultat principal de cette section, nous introduisons un théorème qui concerne les orientations de graphes mixtes, c.-à-d. des graphes ayant aussi bien des arêtes que des arcs. Étant donné un graphe mixte G, peut-on orienter ses arêtes de telle manière que le graphe orienté résultant soit eulérien ? Le théorème suivant répond à cette question :

Théorème 19.18. (Ford et Fulkerson [1962]) *Soit G un graphe orienté et H un graphe non orienté tels que $V(G) = V(H)$. Alors H a une orientation H' telle que le graphe orienté $G + H'$ soit eulérien si et seulement si :*

- $|\delta_G^+(v)| + |\delta_G^-(v)| + |\delta_H(v)|$ *est pair pour tout $v \in V(G)$ et*
- $|\delta_G^+(X)| - |\delta_G^-(X)| \leq |\delta_H(X)|$ *pour tout $X \subseteq V(G)$.*

Preuve. La condition nécessaire est évidente. On prouve la condition suffisante par induction sur $|E(H)|$. Si $E(H) = \emptyset$, l'affirmation est triviale.

On dit qu'un ensemble X est critique si $|\delta_G^+(X)| - |\delta_G^-(X)| = |\delta_H(X)| > 0$. Soit X un ensemble critique. (S'il n'existe pas d'ensemble critique, on oriente arbitrairement une arête et on applique l'hypothèse d'induction.) On choisit une

arête $e \in \delta_H(X)$ et on l'oriente de telle manière que e est entrant dans X. Nous affirmons que les conditions sont encore vérifiées.

Supposons, au contraire, qu'il existe un $Y \subseteq V(G)$ tel que $|\delta_G^+(Y)| - |\delta_G^-(Y)| > |\delta_H(Y)|$. Puisque chaque degré est pair, $|\delta_G^+(Y)| - |\delta_G^-(Y)| - |\delta_H(Y)|$ doit être pair. Cela implique $|\delta_G^+(Y)| - |\delta_G^-(Y)| \geq |\delta_H(Y)| + 2$. Ainsi Y était critique avant l'orientation de e et e sort maintenant de Y.

En appliquant les lemmes 2.1(a) et (b) pour $|\delta_G^+|$ et $|\delta_G^-|$ et le lemme 2.1(c) pour $|\delta_H|$, on obtient (avant l'orientation de e) :

$$\begin{aligned} 0+0 &= |\delta_G^+(X)| - |\delta_G^-(X)| - |\delta_H(X)| + |\delta_G^+(Y)| - |\delta_G^-(Y)| - |\delta_H(Y)| \\ &= |\delta_G^+(X \cap Y)| - |\delta_G^-(X \cap Y)| - |\delta_H(X \cap Y)| \\ &\quad + |\delta_G^+(X \cup Y)| - |\delta_G^-(X \cup Y)| - |\delta_H(X \cup Y)| - 2|E_H(X,Y)| \\ &\leq 0 + 0 - 2|E_H(X,Y)| \leq 0. \end{aligned}$$

On a donc égalité partout et on en conclut que $E_H(X,Y) = \emptyset$, ce qui contredit l'existence de e. □

Corollaire 19.19. *Un graphe eulérien non orienté peut être orienté de telle manière que l'on obtienne un graphe eulérien orienté.* □

Bien entendu ce corollaire peut être prouvé plus facilement en orientant les arêtes selon leur occurrence dans un cycle eulérien.

Revenons maintenant au PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES.

Théorème 19.20. (Rothschild et Whinston [1966]) *Soit (G,H) une instance du* PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES *telle que $G+H$ soit eulérien et que H soit l'union de deux étoiles (c.-à-d. deux sommets rencontrant toutes les arêtes de demande). Alors (G,H) a une solution si et seulement si le critère de coupe est vérifié.*

Preuve. Montrons que le critère de coupe est suffisant. Soient t_1, t_2 deux sommets rencontrant toutes les arêtes de demande. Introduisons d'abord deux nouveaux sommets s_1' et s_2'. On remplace chaque arête de demande (t_1, s_i) par une nouvelle arête de demande (t_1, s_1') et une nouvelle arête d'offre (s_1', s_i). De même, on remplace chaque arête de demande (t_2, s_i) par une nouvelle arête de demande (t_2, s_2') et une nouvelle arête d'offre (s_2', s_i).

L'instance résultante (G', H') est équivalente à (G,H) et H' est seulement constitué de deux ensembles d'arêtes parallèles. Il est facile de voir que le critère de coupe est encore vérifié. De plus, $G' + H'$ est eulérien.

On oriente alors les arêtes de H' arbitrairement, mais de telle façon que les arêtes parallèles aient la même orientation (on note le résultat H''). Les deux graphes H'' et G' vérifient les hypothèses du théorème 19.18, car le critère de coupe implique $|\delta_{H''}^+(X)| - |\delta_{H''}^-(X)| \leq |\delta_{G'}(X)|$ pour tout $X \subseteq V(G)$. Ainsi on peut orienter les arêtes de G' afin d'obtenir un graphe orienté G'' tel que $G'' + H''$ soit eulérien.

On considère (G'', H'') comme une instance du PROBLÈME DES CHEMINS ARC-DISJOINTS. (G'', H'') satisfait le critère de coupe (orienté). Mais alors le

théorème 19.8 garantit une solution qui, en supprimant les orientations, est aussi une solution pour (G', H'). □

Le même théorème est vérifié si H (en négligeant les arêtes parallèles) correspond au graphe K_4 ou C_5 (le cycle de longueur 5) (Lomonosov [1979], Seymour [1980]). Dans le cas du graphe K_5, au moins le critère de distance est suffisant (Karzanov [1987]). Cependant, le problème devient *NP*-difficile si H peut avoir trois ensembles d'arêtes parallèles, comme nous l'avons vu au théorème 19.14.

Exercices

1. Soit (G, H) une instance du PROBLÈME DES CHEMINS ARC-DISJOINTS ou du PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES, violant le critère de distance (19.2) pour une certaine fonction $z : E(G) \to \mathbb{R}_+$. Prouver qu'alors il existe une certaine fonction $z : E(G) \to \mathbb{Z}_+$ violant (19.2). De plus, donner des exemples où il n'existe pas de fonction $z : E(G) \to \{0, 1\}$ violant (19.2).

* 2. Pour une instance (G, H) du PROBLÈME DES CHEMINS ARC-DISJOINTS (ou du PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES), on considère la relaxation du PROBLÈME DU MULTIFLOT et on résout

$$\min \{\lambda : \lambda \in \mathbb{R},\ y \geq 0,\ My \leq \lambda \mathbb{1},\ Ny = \mathbb{1}\},$$

où M et N sont définies comme au lemme 19.1. Soit (y^*, λ^*) une solution optimale. On recherche alors une solution entière, c.-à-d. un chemin de s à t P_f pour chaque arc de demande $f = (t, s) \in E(H)$, telle que la charge maximale sur un arc d'offre soit minimum (la charge d'un arc correspond au nombre de chemins qui l'empruntent). On fait cela par arrondi aléatoire : on choisit indépendamment pour chaque arc de demande un chemin P avec probabilité y_P.
Soit $0 < \epsilon \leq 1$ et supposons que $\lambda^* \geq 3 \ln \frac{|E(G)|}{\epsilon}$. Prouver qu'alors l'arrondi aléatoire précédent fournit une solution entière avec une charge maximale au plus égale à $\lambda^* + \sqrt{3\lambda^* \ln \frac{|E(G)|}{\epsilon}}$, avec une probabilité au moins égale à $1 - \epsilon$.
Indication : utiliser les résultats suivants de la théorie des probabilités : si $B(m, N, p)$ est la probabilité d'avoir au moins m succès parmi N épreuves de Bernoulli indépendantes, ayant chacune une probabilité de succès égale à p, alors

$$B((1 + \beta)Np, N, p) \ < \ e^{-\frac{1}{3}\beta^2 Np}$$

pour tout $0 < \beta \leq 1$. De plus, la probabilité d'avoir au moins m succès parmi N épreuves de Bernoulli indépendantes, ayant des probabilités de succès égales à $p_1, \ldots, p_N$, est au plus égale à $B\left(m, N, \frac{1}{N}(p_1 + \cdots + p_N)\right)$.
(Raghavan et Thompson [1987])

3. Prouver qu'il existe un algorithme polynomial pour le PROBLÈME DES CHEMINS ARC-DISJOINTS ou le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES

lorsque $G + H$ est eulérien et H est seulement constitué de deux ensembles d'arêtes parallèles.

4. Montrer que, dans un graphe orienté donné, on peut trouver en temps polynomial un ensemble minimum d'arcs rencontrant toutes les coupes orientées. Montrer que pour les graphes planaires le nombre arc-transversal des circuits peut être déterminé en temps polynomial.

5. Montrer que, dans un graphe orienté, on peut trouver en temps polynomial un ensemble minimum d'arcs dont la contraction rend le graphe fortement connexe.

6. Prouver que le résultat du corollaire 19.11 n'est pas vrai pour les graphes orientés généraux (non planaires).

7. Montrer que l'affirmation du corollaire 19.12 devient fausse si la condition «G est sans circuits» est omise.
Remarque : dans ce cas le PROBLÈME DES CHEMINS ARC-DISJOINTS est *NP*-difficile (Vygen [1995]).

8. Prouver que le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES peut être résolu en temps polynomial si $G + H$ est planaire et eulérien.

∗ 9. On considère les instances (G, H) du PROBLÈME DES CHAÎNES SOMMET-DISJOINTES où G est planaire et où les nœuds terminaux sont distincts (c.-à-d. $e \cap f = \emptyset$ pour toute paire d'arêtes e et f) et sont situés sur la face extérieure. Soit (G, H) une telle instance, où G est 2-connexe. Soit C le cycle qui borde la face extérieure (voir proposition 2.31).

Prouver que (G, H) a une solution si et seulement si les conditions suivantes sont vérifiées :

a) $G + H$ est planaire.

b) Aucun ensemble $X \subseteq V(G)$ ne sépare plus de $|X|$ arêtes de demande (on dit que X sépare (v, w) si $(v, w) \cap X \neq \emptyset$ ou si w n'est pas connecté à v dans $G - X$).

En conclure que le PROBLÈME DES CHAÎNES SOMMET-DISJOINTES dans les graphes planaires avec des nœuds distincts sur la face extérieure peut être résolu en temps polynomial.
Indication : pour prouver que les conditions (a) et (b) sont suffisantes, considérer l'étape suivante : soit $f = (v, w)$ une arête de demande telle qu'au moins une des deux chaînes de v à w C ne contienne pas d'autre terminal. Réaliser f à l'aide de cette chaîne et le supprimer.

Remarque : Robertson et Seymour [1986] ont étendu cela à une condition nécessaire et suffisante pour l'existence d'une solution du PROBLÈME DES CHAÎNES SOMMET-DISJOINTES avec deux arêtes de demande.

∗ 10. Soit $k \in \mathbb{N}$ fixé. Prouver qu'il existe un algorithme polynomial pour le PROBLÈME DES CHAÎNES SOMMET-DISJOINTES restreint aux graphes de largeur d'arbre au plus égale à k (voir exercice 22 du chapitre 2).
Remarque : Scheffler [1994] a prouvé qu'il existe en fait un algorithme linéaire.

Au contraire, le PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES est *NP*-difficile même pour les graphes de largeur d'arbre égale à 2 (Nishizeki, Vygen et Zhou [2001]).

11. Prouver que le PROBLÈME DES CHEMINS SOMMET-DISJOINTS et le PROBLÈME DES CHAÎNES SOMMET-DISJOINTES sont *NP*-difficiles. Prouver que la partie «sommet-disjointes» du théorème 19.16 implique sa partie «arête-disjointes».

12. Montrer que la preuve du théorème d'Okamura-Seymour mène à un algorithme polynomial.

13. Soit (G, H) une instance du PROBLÈME DES CHAÎNES ARÊTE-DISJOINTES et supposons que G est planaire, que tous les sommets terminaux sont situés sur la face extérieure et que chaque sommet qui n'est pas situé sur la face extérieure a un degré pair. De plus, supposons que

$$|\delta_G(X)| \; > \; |\delta_H(X)| \quad \text{pour tout } X \subseteq V(G).$$

Prouver que (G, H) a une solution.
Indication : utiliser le théorème d'Okamura-Seymour.

14. En généralisant le théorème de Robbins (exercice 17(c) du chapitre 2), formuler et prouver une condition nécessaire et suffisante pour l'existence d'une orientation des arêtes d'un graphe mixte telle que le graphe orienté résultant soit fortement connexe.
(Boesch et Tindell [1980])

15. Soit (G, H) une instance du PROBLÈME DES CHEMINS ARC-DISJOINTS où $G + H$ est eulérien, G est planaire et sans circuit, et tous les sommets terminaux sont situés sur la face extérieure. Prouver que (G, H) a une solution si et seulement si le critère de coupe est vérifié.
Indication : utiliser le lemme 19.13 et le théorème d'Okamura-Seymour 19.17.

16. Prouver le théorème 19.18 en utilisant des techniques de flot.

17. Prouver le théorème d'orientation de Nash-Williams [1969], qui est une généralisation du théorème de Robbins (exercice 17(c) du chapitre 2) :
un graphe non orienté G peut être orienté afin qu'il devienne k-arc-connexe (c.-à-d. qu'il existe k chemins de s à t arc-disjoints pour toute paire $(s, t) \in V(G) \times V(G)$) si et seulement si G est $2k$-arête-connexe.
Indication : pour prouver la condition suffisante, considérons G' une orientation quelconque de G. Prouver que le système

$$\begin{aligned} x_e &\leq 1 && (e \in E(G')), \\ x_e &\geq 0 && (e \in E(G')), \\ \sum_{e \in \delta^-(X)} x_e - \sum_{e \in \delta^+_{G'}(X)} x_e &\leq |\delta^-_{G'}(X)| - k && (\emptyset \neq X \subset V(G')) \end{aligned}$$

est TDI, comme dans la preuve du théorème de Lucchesi-Younger 19.10.
(Frank [1980]), (Frank et Tardos [1984])

18. Prouver le théorème du multiflot à deux-commodités de Hu : une instance (G, H, u, b) du PROBLÈME DU MULTIFLOT NON ORIENTÉ telle que $|E(H)| = 2$ a une solution si et seulement si $\sum_{e\in\delta_G(X)} u(e) \geq \sum_{f\in\delta_H(X)} b(f)$ pour tout $X \subseteq V(G)$, c.-à-d. si et seulement si la condition de coupe est vérifiée.
Indication : utiliser le théorème 19.20.
(Hu [1963])

Références

Littérature générale :

Frank, A. [1990] : Packing paths, circuits and cuts – a survey. In : Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, eds.), Springer, Berlin 1990, pp. 47–100

Ripphausen-Lipa, H., Wagner, D., Weihe, K. [1995] : Efficient algorithms for disjoint paths in planar graphs. In : Combinatorial Optimization ; DIMACS Series in Discrete Mathematics and Theoretical Computer Science 20 (W. Cook, L. Lovász, P. Seymour, eds.), AMS, Providence 1995

Schrijver, A. [2003] : Combinatorial Optimization : Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 70–76

Vygen, J. [1994] : Disjoint Paths. Report No. 94816, Research Institute for Discrete Mathematics, University of Bonn, 1994

Références citées :

Becker, M., Mehlhorn, K. [1986] : Algorithms for routing in planar graphs. Acta Informatica 23 (1986), 163–176

Bienstock, D., Iyengar, G. [2006] : Solving fractional packing problems in $O^*(\frac{1}{\epsilon})$ iterations. SIAM Journal on Computing 35 (2006), 825–854

Boesch, F., Tindell, R. [1980] : Robbins's theorem for mixed multigraphs. American Mathematical Monthly 87 (1980), 716–719

Chudak, F.A., Eleutério, V. [2005] : Improved approximation schemes for linear programming relaxations of combinatorial optimization problems. In : Integer Programming and Combinatorial Optimization ; Proceedings of the 11th International IPCO Conference ; LNCS 3509 (M. Jünger, V. Kaibel, eds.), Springer, Berlin 2005, pp. 81–96

Even, S., Itai, A., Shamir, A. [1976] : On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing 5 (1976), 691–703

Fleischer, L.K. [2000] : Approximating fractional multicommodity flow independent of the number of commodities. SIAM Journal on Discrete Mathematics 13 (2000), 505–520

Ford, L.R., Fulkerson, D.R. [1958] : A suggested computation for maximal multicommodity network flows. Management Science 5 (1958), 97–101

Ford, L.R., Fulkerson, D.R. [1962] : Flows in Networks. Princeton University Press, Princeton 1962

Fortune, S., Hopcroft, J., Wyllie, J. [1980] : The directed subgraph homeomorphism problem. Theoretical Computer Science 10 (1980), 111–121

Frank, A. [1980] : On the orientation of graphs. Journal of Combinatorial Theory B 28 (1980), 251–261

Frank, A. [1981] : How to make a digraph strongly connected. Combinatorica 1 (1981), 145–153

Frank, A., Tardos, É. [1984] : Matroids from crossing families. In : Finite and Infinite Sets ; Vol. I (A. Hajnal, L. Lovász, and V.T. Sós, eds.), North-Holland, Amsterdam, 1984, pp. 295–304

Garg, N., Könemann, J. [1998] : Faster and simpler algorithms for multicommodity flow and other fractional packing problems. Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (1998), 300–309

Grigoriadis, M.D., Khachiyan, L.G. [1996] : Coordination complexity of parallel price-directive decomposition. Mathematics of Operations Research 21 (1996), 321–340

Hu, T.C. [1963] : Multi-commodity network flows. Operations Research 11 (1963), 344–360

Ibaraki, T., Poljak, S. [1991] : Weak three-linking in Eulerian digraphs. SIAM Journal on Discrete Mathematics 4 (1991), 84–98

Karakostas, G. [2002] : Faster approximation schemes for fractional multicommodity flow problems. Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (2002), 166–173

Karp, R.M. [1972] : Reducibility among combinatorial problems. In : Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, eds.), Plenum Press, New York 1972, pp. 85–103

Karzanov, A.V. [1987] : Half-integral five-terminus-flows. Discrete Applied Mathematics 18 (1987) 263–278

Lomonosov, M. [1979] : Multiflow feasibility depending on cuts. Graph Theory Newsletter 9 (1979), 4

Lovász, L. [1976] : On two minimax theorems in graph. Journal of Combinatorial Theory B 21 (1976), 96–103

Lucchesi, C.L., Younger, D.H. [1978] : A minimax relation for directed graphs. Journal of the London Mathematical Society II 17 (1978), 369–374

Matsumoto, K., Nishizeki, T., Saito, N. [1986] : Planar multicommodity flows, maximum matchings and negative cycles. SIAM Journal on Computing 15 (1986), 495–510

Middendorf, M., Pfeiffer, F. [1993] : On the complexity of the disjoint path problem. Combinatorica 13 (1993), 97–107

Nash-Williams, C.S.J.A. [1969] : Well-balanced orientations of finite graphs and unobtrusive odd-vertex-pairings. In : Recent Progress in Combinatorics (W. Tutte, ed.), Academic Press, New York 1969, pp. 133–149

Nishizeki, T., Vygen, J., Zhou, X. [2001] : The edge-disjoint paths problem is *NP*-complete for series-parallel graphs. Discrete Applied Mathematics 115 (2001), 177–186

Okamura, H., Seymour, P.D. [1981] : Multicommodity flows in planar graphs. Journal of Combinatorial Theory B 31 (1981), 75–81

Raghavan, P., Thompson, C.D. [1987] : Randomized rounding : a technique for provably good algorithms and algorithmic proofs. Combinatorica 7 (1987), 365–374

Robertson, N., Seymour, P.D. [1986] : Graph minors VI ; Disjoint paths across a disc. Journal of Combinatorial Theory B 41 (1986), 115–138

Robertson, N., Seymour, P.D. [1995] : Graph minors XIII ; The disjoint paths problem. Journal of Combinatorial Theory B 63 (1995), 65–110

Rothschild, B., Whinston, A. [1966] : Feasibility of two-commodity network flows. Operations Research 14 (1966), 1121–1129

Scheffler, P. [1994] : A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Technical Report No. 396/1994, FU Berlin, Fachbereich 3 Mathematik

Seymour, P.D. [1981] : On odd cuts and multicommodity flows. Proceedings of the London Mathematical Society (3) 42 (1981), 178–192

Shahrokhi, F., Matula, D.W. [1990] : The maximum concurrent flow problem. Journal of the ACM 37 (1990), 318–334

Shmoys, D.B. [1996] : Cut problems and their application to divide-and-conquer. In : Approximation Algorithms for *NP*-Hard Problems (D.S. Hochbaum, ed.), PWS, Boston, 1996

Vygen, J. [1995] : *NP*-completeness of some edge-disjoint paths problems. Discrete Applied Mathematics 61 (1995), 83–90

Vygen, J. [2004] : Near-optimum global routing with coupling, delay bounds, and power consumption. In : Integer Programming and Combinatorial Optimization ; Proceedings of the 10th International IPCO Conference ; LNCS 3064 (G. Nemhauser, D. Bienstock, eds.), Springer, Berlin 2004, pp. 308–324

Wagner, D., Weihe, K. [1995] : A linear-time algorithm for edge-disjoint paths in planar graphs. Combinatorica 15 (1995), 135–150

Young, N. [1995] : Randomized rounding without solving the linear program. Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (1995), 170–178

Chapitre 20

Problèmes de conception de réseaux

La connexité est une notion très importante de l'optimisation combinatoire. Au chapitre 8, nous avons montré comment calculer la connexité entre chaque paire de sommets d'un graphe non orienté. Nous recherchons maintenant des sous-graphes qui vérifient certaines conditions de connexité. Le problème général est le suivant :

PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE	
Instance	Un graphe non orienté G avec des poids $c : E(G) \to \mathbb{R}_+$ et une condition de connexité $r_{xy} \in \mathbb{Z}_+$ pour chaque paire (non ordonnée) de sommets x, y.
Tâche	Trouver un sous-graphe couvrant de poids minimum H de G tel que, pour chaque paire de sommets x, y, il existe au moins r_{xy} chaînes arête-disjointes de x à y dans H.

Des applications pratiques de ce problème apparaissent par exemple dans la conception de réseaux de télécommunication pouvant «survivre» à la défaillance de certains liens.

On peut aussi supposer que les arêtes peuvent être sélectionnées un nombre quelconque de fois (voir Goemans et Bertsimas [1993], Bertsimas et Teo [1997]). Cette variante est cependant un cas particulier du problème général puisque G peut avoir de nombreuses arêtes parallèles.

Aux paragraphes 20.1 et 20.2, on considère d'abord le PROBLÈME DE L'ARBRE DE STEINER, qui est un cas particulier bien connu. Dans ce problème, on a un ensemble $T \subseteq V(G)$ de sommets dits terminaux tel que $r_{xy} = 1$ si $x, y \in T$ et $r_{xy} = 0$ sinon. On recherche un réseau de longueur minimum reliant tous les terminaux ; un tel réseau est appelé un connecteur et un connecteur minimum est un arbre de Steiner :

Définition 20.1. *Soit G un graphe non orienté et $T \subseteq V(G)$. Un* **connecteur** *pour T est un graphe connexe Y tel que $T \subseteq V(Y)$. Un* **arbre de Steiner** *pour T dans G est un arbre S tel que $T \subseteq V(S) \subseteq V(G)$ et $E(S) \subseteq E(G)$. Les éléments de T sont appelés des* **terminaux***, ceux de $V(S) \setminus T$ sont les* **points de Steiner** *de S.*

Il est parfois également requis que toutes les feuilles d'un arbre de Steiner soient des terminaux. Cela peut évidemment toujours être réalisé en supprimant si besoin des arêtes.

Au paragraphe 20.3, nous passons au PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE général et nous donnons deux algorithmes d'approximation aux paragraphes 20.4 et 20.5. Bien que le premier algorithme soit le plus rapide, on peut garantir avec le second un rapport de performance égal à 2 en temps polynomial.

20.1 Arbres de Steiner

Nous considérons dans ce paragraphe le problème suivant :

PROBLÈME DE L'ARBRE DE STEINER	
Instance	Un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$ et un ensemble $T \subseteq V(G)$.
Tâche	Trouver un arbre de Steiner S pour T dans G dont le poids $c(E(S))$ soit minimum.

Nous avons déjà traité les cas particuliers $T = V(G)$ (arbre couvrant) et $|T| = 2$ (plus courte chaîne) aux chapitres 6 et 7. Alors que nous avions un algorithme polynomial dans ces deux cas, le problème général est *NP*-difficile.

Théorème 20.2. (Karp [1972]) *Le* PROBLÈME DE L'ARBRE DE STEINER *est NP-difficile, même pour des poids unitaires.*

Preuve. Nous décrivons une transformation à partir du problème de COUVERTURE PAR LES SOMMETS qui est *NP*-complet d'après le corollaire 15.24. Étant donné un graphe G, on considère le graphe H constitué des sommets $V(H) := V(G) \dot\cup E(G)$ et des arêtes (v, e) pour $v \in e \in E(G)$ et (v, w) pour $v, w \in V(G)$, $v \neq w$. (Voir figure 20.1 pour une illustration.) On pose $c(e) := 1$ pour tout $e \in E(H)$ et $T := E(G)$.

Étant donné une couverture par les sommets $X \subseteq V(G)$ de G, on peut connecter X dans H à l'aide d'un arbre ayant $|X|-1$ arêtes et relier chacun des sommets de T par une arête. On obtient un arbre de Steiner avec $|X| - 1 + |E(G)|$ arêtes. D'autre part, soit $(T \cup X, F)$ un arbre de Steiner pour T dans H. Alors X est une couverture par les sommets dans G et $|F| = |T \cup X| - 1 = |X| + |E(G)| - 1$.

Ainsi G a une couverture par les sommets de cardinalité k si et seulement si H a un arbre de Steiner pour T avec $k + |E(G)| - 1$ arêtes. □

Cette transformation fournit également le résultat plus fort suivant :

Figure 20.1.

Théorème 20.3. (Bern et Plassmann [1989]) *Le* PROBLÈME DE L'ARBRE DE STEINER *est MAXSNP-difficile, même pour des poids unitaires.*

Preuve. La transformation de la preuve précédente n'est pas en général une L-réduction, mais nous affirmons que cela en est une si G est de degré borné. D'après le théorème 16.46, le PROBLÈME DE COUVERTURE MINIMUM PAR LES SOMMETS pour les graphes de degré maximum égal à 4 est *MAXSNP*-difficile.

Pour chaque arbre de Steiner $(T \cup X, F)$ dans H et la couverture par les sommets correspondante X dans G nous avons

$$\begin{aligned} |X| - \mathrm{OPT}(G) &= (|F| - |E(G)| + 1) - (\mathrm{OPT}(H,T) - |E(G)| + 1) \\ &= |F| - \mathrm{OPT}(H,T). \end{aligned}$$

De plus, $\mathrm{OPT}(H,T) \leq 2|T| - 1 = 2|E(G)| - 1$ et $\mathrm{OPT}(G) \geq \frac{|E(G)|}{4}$ si G est de degré maximum égal à 4. Ainsi $\mathrm{OPT}(H,T) < 8\,\mathrm{OPT}(G)$ et nous concluons que la transformation est en effet une L-réduction. □

Deux autres variantes du PROBLÈME DE L'ARBRE DE STEINER sont également *NP*-difficiles : le PROBLÈME DE L'ARBRE DE STEINER EUCLIDIEN (Garey, Graham et Johnson [1977]) et le PROBLÈME DE L'ARBRE DE STEINER DE MANHATTAN (Garey et Johnson [1977]). Dans ces deux problèmes, on recherche un réseau (un ensemble de segments) de longueur totale minimum qui connecte un ensemble donné de points dans le plan. La différence entre ces deux problèmes est que l'on ne considère que des segments horizontaux et verticaux dans le PROBLÈME DE L'ARBRE DE STEINER DE MANHATTAN. Contrairement au PROBLÈME DE L'ARBRE DE STEINER dans les graphes qui est *MAXSNP*-difficile, ces deux versions géométriques ont un schéma d'approximation. Une variante de cet algorithme, qui est due à Arora [1998], résout également le PVC EUCLIDIEN et certains autres problèmes géométriques et sera présentée au paragraphe 21.2 (voir exercice 8 du chapitre 21). Le PROBLÈME DE L'ARBRE DE STEINER dans les graphes planaires a également un schéma d'approximation, trouvé par Borradaile, Kenyon-Mathieu et Klein [2007].

Hanan [1966] a montré que le PROBLÈME DE L'ARBRE DE STEINER DE MANHATTAN peut être réduit au PROBLÈME DE L'ARBRE DE STEINER dans les graphes qui sont des grilles finies : il existe toujours une solution optimale où tous les segments sont situés sur la grille induite par les coordonnées des terminaux. Le

PROBLÈME DE L'ARBRE DE STEINER DE MANHATTAN est important pour les problèmes d'intégration à très grande échelle (en anglais *VLSI design*), où des composants électroniques doivent être connectés à l'aide de fils horizontaux et verticaux ; voir Korte, Prömel et Steger [1990], Martin [1992] et Hetzel [1995]. On recherche alors plusieurs arbres de Steiner disjoints. Cela est une généralisation du PROBLÈME DES CHAÎNES DISJOINTES présenté au chapitre 19.

Nous allons maintenant décrire un algorithme de programmation dynamique dû à Dreyfus et Wagner [1972]. Cet algorithme résout le PROBLÈME DE L'ARBRE DE STEINER de manière exacte, mais a un temps de calcul exponentiel en général.

L'ALGORITHME DE DREYFUS-WAGNER calcule un arbre de Steiner optimum pour chaque sous-ensemble de T, en commençant avec les ensembles de deux éléments. Il utilise la formule récursive suivante :

Lemme 20.4. *Soit (G, c, T) une instance du* PROBLÈME DE L'ARBRE DE STEINER. *Pour chaque $U \subseteq T$ et $x \in V(G) \setminus U$, on définit*

$$\begin{aligned} p(U) &:= \min\{c(E(S)) : S \text{ est un arbre de Steiner pour } U \text{ dans } G\};\\ q(U \cup \{x\}, x) &:= \min\{c(E(S)) : S \text{ est un arbre de Steiner pour } U \cup \{x\}\\ &\qquad\qquad \text{dans } G \text{ dont les feuilles sont des éléments de } U\}. \end{aligned}$$

On a alors pour tout $U \subseteq V(G)$, $|U| \geq 2$ et $x \in V(G) \setminus U$:

(a) $q(U \cup \{x\}, x) \;=\; \min_{\emptyset \neq U' \subset U} \Big(p(U' \cup \{x\}) + p((U \setminus U') \cup \{x\})\Big)$,

(b) $p(U \cup \{x\}) \;=\; \min\Big\{ \min_{y \in U} \Big(p(U) + \mathrm{dist}_{(G,c)}(x,y)\Big),$
$\min_{y \in V(G)\setminus U} \Big(q(U \cup \{y\}, y) + \mathrm{dist}_{(G,c)}(x,y)\Big) \Big\}$.

Preuve. (a) : Tout arbre de Steiner S pour $U \cup \{x\}$ dont les feuilles sont des éléments de U est l'union disjointe de deux arbres, chacun contenant x et au moins un élément de U. On obtient donc l'équation (a).

(b) : L'inégalité $\leq$ est évidente. Considérons un arbre de Steiner optimum S pour $U \cup \{x\}$. Si $|\delta_S(x)| \geq 2$, alors

$$p(U \cup \{x\}) \;=\; c(E(S)) \;=\; q(U \cup \{x\}, x) \;=\; q(U \cup \{x\}, x) + \mathrm{dist}_{(G,c)}(x,x).$$

Si $|\delta_S(x)| = 1$, alors soit y le sommet le plus proche de x dans S qui appartient à U ou vérifie $|\delta_S(y)| \geq 3$. Nous distinguons deux cas : si $y \in U$, alors

$$p(U \cup \{x\}) \;=\; c(E(S)) \;\geq\; p(U) + \mathrm{dist}_{(G,c)}(x,y),$$

sinon

$$p(U \cup \{x\}) \;=\; c(E(S)) \;\geq\; \min_{y \in V(G)\setminus U} \big(q(U \cup \{y\}, y) + \mathrm{dist}_{(G,c)}(x,y)\big)\,.$$

Dans (b), on calcule le minimum sur ces trois formules. □

Ces formules récursives suggèrent immédiatement l'algorithme de programmation dynamique suivant :

ALGORITHME DE DREYFUS-WAGNER

Input Un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$ et un ensemble $T \subseteq V(G)$.

Output La longueur $p(T)$ d'un arbre de Steiner optimum pour T dans G.

① **If** $|T| \leq 1$ **then** poser $p(T) := 0$ et **stop**.
Calculer $\text{dist}_{(G,c)}(x, y)$ pour tout $x, y \in V(G)$.
Poser $p(\{x, y\}) := \text{dist}_{(G,c)}(x, y)$ pour tout $x, y \in V(G)$.

② **For** $k := 2$ **to** $|T| - 1$ **do** :
For tout $U \subseteq T$ tel que $|U| = k$ et tout $x \in V(G) \setminus U$ **do** :
Poser $q(U \cup \{x\}, x) := \min_{\emptyset \neq U' \subset U} \Big(p(U' \cup \{x\}) + p((U \setminus U') \cup \{x\})\Big)$.
For tout $U \subseteq T$ tel que $|U| = k$ et tout $x \in V(G) \setminus U$ **do** :

$$\text{Poser } p(U \cup \{x\}) := \min \Bigg\{ \min_{y \in U} \big(p(U) + \text{dist}_{(G,c)}(x, y)\big), \min_{y \in V(G) \setminus U} \big(q(U \cup \{y\}, y) + \text{dist}_{(G,c)}(x, y)\big) \Bigg\}.$$

Théorème 20.5. (Dreyfus et Wagner [1972]) *L'*ALGORITHME DE DREYFUS-WAGNER *détermine correctement la longueur d'un arbre de Steiner optimum en temps* $O\big(3^t n + 2^t n^2 + mn + n^2 \log n\big)$, *où* $n = |V(G)|$ *et* $t = |T|$.

Preuve. La validité de l'algorithme est une conséquence du lemme 20.4. L'étape ① revient à résoudre le PROBLÈME DU PLUS COURT CHEMIN POUR TOUTES LES PAIRES DE SOMMETS, ce qui peut être fait en temps $O(mn + n^2 \log n)$ d'après le théorème 7.9.

La première récursion de l'étape ② requiert un temps $O\left(3^t n\right)$ puisqu'il existe 3^t possibilités de partitionner T en $U', U \setminus U'$ et $T \setminus U$. La seconde récursion de l'étape ② requiert évidemment un temps $O\left(2^t n^2\right)$. □

Sous cette forme, l'ALGORITHME DE DREYFUS-WAGNER calcule la longueur d'un arbre de Steiner optimum, mais pas l'arbre de Steiner lui-même. Cependant, cela peut facilement être réalisé en stockant une information supplémentaire et en procédant par «*backtracking*». Nous avons déjà présenté cela en détail pour l'ALGORITHME DE DIJKSTRA (paragraphe 7.1).

Remarquons que l'algorithme requiert en général un temps exponentiel et un espace mémoire exponentiel. Pour un nombre borné de terminaux, c'est un algorithme en $O(n^3)$. Il existe un autre cas particulier intéressant où il s'exécute en temps polynomial (et en espace polynomial) : si G est un graphe planaire et si tous les terminaux sont situés sur la face extérieure, on peut alors modifier l'ALGORITHME DE DREYFUS-WAGNER pour obtenir une complexité $O\left(n^3 t^2\right)$ (exercice 3). Dans le cas général (avec un nombre quelconque de terminaux), le temps de calcul a été amélioré jusqu'à $O\Big(\big(2 + \frac{1}{d}\big)^t n^{12\sqrt{d \ln d}}\Big)$ pour d suffisamment grand par Fuchs *et al.* [2007].

Comme on ne peut espérer avoir un algorithme polynomial exact pour le PROBLÈME DE L'ARBRE DE STEINER général, il est intéressant de considérer des algorithmes d'approximation. L'idée directrice de certains de ces algorithmes est d'approximer l'arbre de Steiner optimum pour T dans G par un arbre couvrant de poids minimum dans le sous-graphe de la fermeture métrique de G induit par T.

Théorème 20.6. *Soit G un graphe connexe avec des poids $c : E(G) \to \mathbb{R}_+$ et soit $(\bar{G}, \bar{c})$ la fermeture métrique associée. Soit $T \subseteq V(G)$. Si S est un arbre de Steiner optimum pour T dans G et si M est un arbre couvrant de poids minimum dans $\bar{G}[T]$ (respectivement à $\bar{c}$), alors $\bar{c}(E(M)) \leq 2c(E(S))$.*

Preuve. Considérons le graphe H contenant deux copies de chaque arête de S. H est eulérien, donc d'après le théorème 2.24 il existe un parcours eulérien W dans H. La première apparition des éléments de T dans W définit un ordre sur T et ainsi un tour W' dans $\bar{G}[T]$. Comme $\bar{c}$ vérifie l'inégalité triangulaire ($\bar{c}((x,z)) \leq \bar{c}((x,y)) + \bar{c}((y,z)) \leq c((x,y)) + c((y,z))$ pour tout x, y, z),

$$\bar{c}(W') \;\leq\; c(W) \;=\; c(E(H)) \;=\; 2c(E(S)).$$

Comme W' contient un arbre couvrant de $\bar{G}[T]$ (il suffit d'enlever une arête), le théorème est prouvé. □

Ce théorème a été publié par Gilbert et Pollak [1968] (se référant à E.F. Moore), Choukhmane [1978], Kou, Markowsky et Berman [1981] et Takahashi et Matsuyama [1980]. Il suggère immédiatement l'algorithme d'approximation de facteur 2 suivant :

ALGORITHME DE KOU-MARKOWSKY-BERMAN	
Input	Un graphe connexe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$ et un ensemble $T \subseteq V(G)$.
Output	Un arbre de Steiner pour T dans G.

① Calculer la fermeture métrique $(\bar{G}, \bar{c})$ et une plus courte chaîne P_{st} pour toute paire $s, t \in T$.

② Trouver un arbre couvrant de poids minimum M dans $\bar{G}[T]$ (respectivement à $\bar{c}$).

Poser $E(S) := \bigcup_{(x,y)\in E(M)} E(P_{xy})$ et $V(S) := \bigcup_{(x,y)\in E(M)} V(P_{xy})$.

③ Retourner un sous-graphe couvrant connexe minimal de S.

Théorème 20.7. (Kou, Markowsky et Berman [1981]) *L'*ALGORITHME DE KOU-MARKOWSKY-BERMAN *est un algorithme d'approximation de facteur 2 pour le* PROBLÈME DE L'ARBRE DE STEINER *et s'exécute en temps $O\left(n^3\right)$, où $n = |V(G)|$.*

Preuve. L'exactitude de l'algorithme et la garantie de performance sont des conséquences directes du théorème 20.6. L'étape ① correspond à la résolution d'un PROBLÈME DU PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES DE SOMMETS, qui peut s'effectuer en temps $O\left(n^3\right)$ (théorème 7.9, corollaire 7.11). L'étape ② peut être réalisée en temps $O\left(n^2\right)$ en utilisant l'ALGORITHME DE PRIM (théorème 6.5). L'étape ③ peut être implémentée en BFS pour s'exécuter en temps $O(n^2)$. □

Mehlhorn [1988] et Kou [1990] ont proposé une implémentation de cet algorithme en $O\left(n^2\right)$. L'idée est de calculer, à la place de $\bar{G}[T]$, un graphe similaire dont les arbres couvrants de poids minimum soient également des arbres couvrants de poids minimum de $\bar{G}[T]$.

Un arbre couvrant de poids minimum fournit une approximation de facteur 2 pour toute instance métrique du PROBLÈME DE L'ARBRE DE STEINER. On ne connaissait pas d'algorithme ayant un meilleur rapport de performance avant que Zelikovsky [1993] propose un algorithme d'approximation de facteur $\frac{11}{6}$ pour le PROBLÈME DE L'ARBRE DE STEINER. Le rapport de performance a été par la suite amélioré à $1,75$ par Berman et Ramaiyer [1994], à $1,65$ par Karpinski et Zelikovsky [1997], à $1,60$ par Hougardy et Prömel [1999] et à $1+\frac{\ln 3}{2} \approx 1,55$ par Robins et Zelikovsky [2005]. Cet algorithme, qui est le meilleur connu actuellement, sera présenté au paragraphe suivant.

D'autre part, d'après le théorème 20.3 et le corollaire 16.40, il ne peut exister de schéma d'approximation à moins que *P = NP*. En effet, Clementi et Trevisan [1999] ont montré que, à moins que *P = NP*, il n'existe pas d'algorithme d'approximation de facteur $1,0006$ pour le PROBLÈME DE L'ARBRE DE STEINER. Voir également Thimm [2003].

Un algorithme calculant des arbres de Steiner optimum et qui est assez efficace, en particulier pour le PROBLÈME DE L'ARBRE DE STEINER DE MANHATTAN, a été développé par Warme, Winter et Zachariasen [2000].

20.2 Algorithme de Robins-Zelikovsky

Définition 20.8. *Un* **arbre de Steiner complet** *pour un ensemble de terminaux T dans un graphe G est un arbre Y dans G tel que T soit l'ensemble des feuilles de Y. Tout arbre de Steiner minimal pour T peut être décomposé en ses* **composantes complètes** *qui sont des arbres de Steiner complets pour des sous-ensembles de T. Les unions de composantes complètes ayant chacune au plus k terminaux sont dites* ***k*-restreintes** *(respectivement à l'ensemble donné des terminaux). Plus précisément, un graphe Y est dit k-restreint (dans G respectivement à T) s'il existe des arbres de Steiner complets Y_i pour $T \cap V(Y_i)$ dans G tels que $|T \cap V(Y_i)| \leq k$ ($i = 1, \ldots, t$), que $V(Y) = \bigcup_{i=1}^{t} V(Y_i)$ et que $E(Y)$ soit l'union disjointe des ensembles $E(Y_i)$. Remarquons que des arêtes parallèles peuvent apparaître.*

On définit le ***k*-Steiner ratio** *par*

$$\rho_k := \sup_{(G,c,T)} \left\{ \frac{\min\{c(E(Y)) : Y \text{ connecteur } k\text{-restreint pour } T\}}{\min\{c(E(Y)) : Y \text{ arbre de Steiner pour } T\}} \right\},$$

où le supremum est pris sur toutes les instances du PROBLÈME DE L'ARBRE DE STEINER.

Par exemple, les connecteurs 2-restreints sont composés de chaînes reliant des terminaux. Les connecteurs 2-restreints optimaux pour T dans (G, c) correspondent donc à des arbres couvrants de poids minimum dans $(\bar{G}[T], \bar{c})$; ainsi $\rho_2 \leq 2$ d'après le théorème 20.6. Les graphes étoiles avec des poids unitaires montrent qu'en fait $\rho_2 = 2$ (et qu'en général $\rho_k \geq \frac{k}{k-1}$). Si l'on restreint le supremum aux instances du PROBLÈME DE L'ARBRE DE STEINER DE MANHATTAN, le ratio est meilleur, par exemple $\frac{3}{2}$ pour $k = 2$ (Hwang [1976]).

Théorème 20.9. (Du, Zhang et Feng [1991]) $\rho_{2^s} \leq \frac{s+1}{s}$.

Preuve. Soit (G, c, T) une instance et Y un arbre de Steiner optimum. Sans perte de généralité, supposons que Y soit un arbre de Steiner complet (sinon on traite les composantes complètes séparément). De plus, en dupliquant les sommets et en ajoutant des arêtes de longueur égale à zéro, on peut supposer que Y est un arbre binaire complet dont les feuilles sont les terminaux. Un certain sommet, la racine, est de degré 2 et tous les autres points de Steiner sont de degré 3. On dit qu'un sommet $v \in V(Y)$ est au niveau i si sa distance par rapport à la racine est égale à i. Tous les terminaux sont au même niveau h (la hauteur de l'arbre binaire).

On définit s connecteurs 2^s-restreints pour T, qui ont une longueur totale égale à au plus $(s+1)c(E(Y))$. Pour $v \in V(Y)$, considérons une chaîne $P(v)$ dans Y de v à une certaine feuille, de telle manière que toutes ces chaînes soient deux à deux arête-disjointes (par exemple, descendre une fois à gauche, puis toujours continuer à droite).

Pour $i = 1, \ldots, s$, soit Y_i l'union des composantes complètes suivantes :

- le sous-arbre de Y induit par les sommets jusqu'au niveau i, plus $P(v)$ pour chaque sommet v situé au niveau i ;
- pour chaque sommet u situé au niveau $ks + i$: le sous-arbre induit par les successeurs de u jusqu'au niveau $(k+1)s + i$, plus $P(v)$ pour chaque sommet v du sous-arbre situé au niveau $(k+1)s + i$ $(k = 0, \ldots, \lfloor \frac{h-1-i}{s} \rfloor - 1)$;
- pour chaque sommet u situé au niveau $\lfloor \frac{h-1-i}{s} \rfloor s + i$: le sous-arbre induit par tous les successeurs de u.

Évidemment, chacun de ces arbres est 2^s-restreint et l'union des arbres de Y_i est Y, c.-à-d. est un connecteur pour T. De plus, chaque arête de Y est contenue une fois par chaque ensemble Y_i, sans compter l'apparition dans une chaîne $P(v)$. De plus, chaque chaîne $P(v)$ est utilisée dans seulement un Y_i. Ainsi chaque arête apparaît au plus $s + 1$ fois. □

En particulier, $\rho_k \to 1$ lorsque $k \to \infty$. On ne peut donc pas espérer trouver un connecteur k-restreint optimum en temps polynomial pour k fixé. En effet, ce problème est *NP*-difficile pour tout $k \geq 4$ fixé (Garey et Johnson [1977]). La borne du théorème 20.9 est serrée : Borchers et Du [1997] ont prouvé que $\rho_k = \frac{(s+1)2^s + t}{s2^s + t}$ pour tout $k \geq 2$, où $k = 2^s + t$ et $0 \leq t < 2^s$.

Nous allons présenter un algorithme qui débute avec un arbre couvrant minimum du sous-graphe de la fermeture métrique induit par T et essaye de l'améliorer en utilisant des arbres de Steiner k-restreints complets. Cependant, l'algorithme inclut seulement au plus la moitié d'un tel arbre de Steiner : sa «perte». Pour chaque arbre de Steiner Y, on définit une **perte** de Y comme un ensemble d'arêtes F de coût minimum reliant chaque point de Steiner de degré au moins égal à 3 à un terminal. Voir la figure 20.2 pour un exemple d'arbre de Steiner complet avec une perte (les arêtes en gras), où l'on suppose que le coût d'une arête est proportionnel à sa longueur.

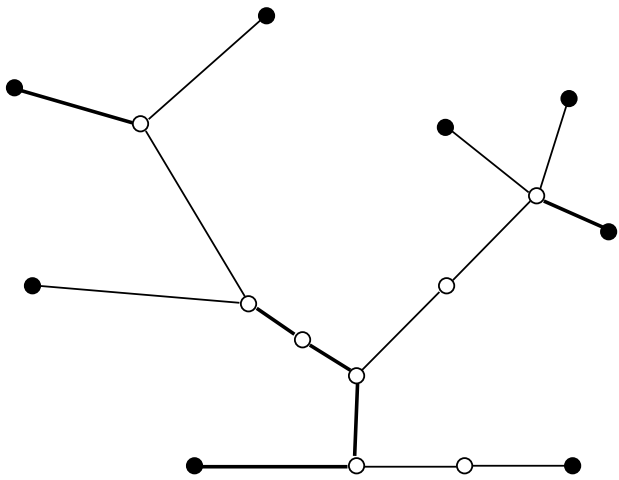


Figure 20.2.

Proposition 20.10. *Soit Y un arbre de Steiner complet pour T, $c : E(Y) \to \mathbb{R}_+$, et soit F une perte de Y. Alors $c(F) \leq \frac{1}{2}c(E(Y))$.*

Preuve. Soit r un sommet quelconque de Y. Soit U l'ensemble des points de Steiner $v \in V(Y) \setminus T$ de degré au moins égal à 3. Pour chaque $v \in U$, on considère une chaîne $P(v)$ de coût minimum parmi les (au moins deux) chaînes allant de v à un sommet $w \in U \cup T$ telle que w soit à une distance plus grande de r que de v. L'union des ensembles d'arêtes de ces chaînes relie chaque élément de U à un terminal et a un coût au plus égal à la moitié du coût total. □

Au lieu de contracter explicitement les pertes des arbres de Steiner k-restreints complets, on ajuste les coûts de la manière suivante :

Proposition 20.11. *Soient G un graphe complet, $T \subseteq V(G)$, $c : E(G) \to \mathbb{R}_+$ et $k \geq 2$. Soit $S \subseteq T$ tel que $|S| \leq k$, soient Y un arbre de Steiner pour S dans G et L une perte de Y. Soit $c'(e) := 0$ pour $e \in L$ et $c'(e) := c(e)$ sinon. On définit*

$c/(Y,L) : E(G) \to \mathbb{R}_+$ par $c/(Y,L)((v,w)) := \min\{c((v,w)), \mathrm{dist}_{(Y,c')}(v,w)\}$ pour $v, w \in S$, $v \neq w$ et $c/(Y,L)(e) := c(e)$ pour toutes les autres arêtes.

Alors il existe un arbre couvrant M dans $G[S]$ tel que $c/(Y,L)(E(M)) + c(L) \leq c(E(Y))$.

De plus, pour chaque connecteur k-restreint H' de T dans G, il existe un connecteur k-restreint H de T dans G tel que $c(E(H)) \leq c/(Y,L)(E(H')) + c(L)$.

Preuve. On prouve la première affirmation par induction sur $|E(Y)|$. On peut supposer que Y est un arbre de Steiner complet (sinon on considère les composantes complètes séparément) et que $|S| > 2$. Alors $L \neq \emptyset$ et il existe un terminal v incident à une arête $e = (v,w) \in L$. En appliquant l'hypothèse d'induction à $Y' := Y - e$ et $(S \setminus \{v\}) \cup \{w\}$ on obtient un arbre couvrant M' tel que $c/(Y', L \setminus \{e\})(M') \leq c(E(Y')) - c(L \setminus \{e\}) = c(E(Y)) - c(L)$. Remplacer w par v dans M ne change pas le coût puisque $c'(e) = 0$.

Pour la seconde affirmation, soit H' un connecteur k-restreint de T. Remplaçons chaque arête $e = (v,w) \in E(H')$ telle que $c/(Y,L)(e) < c(e)$ par une plus courte chaîne de v à w dans (Y,c') et éliminons les arêtes parallèles. Alors le graphe résultant H est un connecteur k-restreint de T et vérifie $c(E(H)) = c'(E(H)) + c(E(H) \cap L) \leq c/(Y,L)(E(H')) + c(L)$. □

Nous modifierons de manière répétée la fonction coût en ajoutant des arêtes qui correspondent aux composantes complètes. L'observation suivante justifie que la réduction du coût d'un arbre couvrant minimum n'augmente pas lorsque d'autres arêtes sont ajoutées auparavant :

Lemme 20.12. (Zelikovsky [1993], Berman et Ramaiyer [1994]) *Soient G un graphe, $c : E(G) \to \mathbb{R}_+$, $T \subseteq V(G)$, (T,U) un autre graphe, $c' : U \to \mathbb{R}_+$. Soit $m : 2^U \to \mathbb{R}_+$, où $m(X)$ est le coût d'un arbre couvrant minimum dans $(T, E(G[T]) \cup X)$. Alors m est supermodulaire.*

Preuve. Soient $A \subseteq U$ et $f \in U$. On exécute l'ALGORITHME DE KRUSKAL en parallèle sur $G[T]$ et sur $G[T] + f$, en considérant les arêtes de $G[T]$ dans le même ordre (par poids non décroissants). Les deux versions s'exécutent exactement de la même façon, sauf que la première ne choisit pas f, alors que la seconde ne choisit pas la première arête qui crée un cycle dans $G + f$ contenant f. Ainsi les coûts minimum des arbres couvrants des deux graphes diffèrent de $\min\{\gamma : G[T] + f$ contient un cycle contenant f dont les arêtes ont un coût au plus égal à $\gamma\} - c'(f)$. Évidemment, cette différence ne peut que diminuer si on considère $G[T] + A$ et $(G[T] + A) + f$ au lieu de $G[T]$ et $G[T] + f$. Ainsi

$$m(A) - m(A \cup \{f\}) \;\leq\; m(\emptyset) - m(\{f\}).$$

Soient maintenant $X, Y \subseteq U$, $Y \setminus X = \{y_1, \ldots, y_k\}$ et notons $m_i(Z) := m((X \cap Y) \cup \{y_1, \ldots, y_{i-1}\} \cup Z)$ pour $i = 1, \ldots, k$. En appliquant le résultat précédent à m_i, on obtient

$$\begin{aligned} m(X) - m(X \cup Y) &= \sum_{i=1}^{k} (m_i(X \setminus Y) - m_i((X \setminus Y) \cup \{y_i\})) \\ &\leq \sum_{i=1}^{k} (m_i(\emptyset) - m_i(\{y_i\})) \\ &= m(X \cap Y) - m(Y), \end{aligned}$$

c.-à-d. la supermodularité de m. □

Nous décrivons maintenant l'algorithme. Notons $mst(c)$ le coût minimum d'un arbre couvrant dans le sous-graphe de $(\bar{G}, c)$ induit par T.

ALGORITHME DE ROBINS-ZELIKOVSKY

Input Un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$ et un ensemble $T \subseteq V(G)$ de terminaux. Un nombre $k \geq 2$.

Output Un arbre de Steiner pour T dans G.

① Calculer la fermeture métrique $(\bar{G}, \bar{c})$ de (G, c).

② Choisir un sous-ensemble S d'au plus k terminaux et une paire $K = (Y, L)$, où Y est un arbre de Steiner optimum pour S et L est une perte de Y, telle que $\frac{mst(\bar{c}) - mst(\bar{c}/K)}{\bar{c}(L)}$ soit maximum et au moins égal à 1.
If un tel choix est impossible, **then go to** ④.

③ Poser $\bar{c} := \bar{c}/K$.
Go to ②.

④ Calculer un arbre couvrant minimum dans le sous-graphe de $(\bar{G}, \bar{c})$ induit par T.
Remplacer toutes les arêtes par des plus courtes chaînes dans (G, c'), où $c'(e) := 0$ si $e \in L$ pour un certain L choisi dans l'algorithme et $c'(e) = c(e)$ sinon.
Finalement calculer un sous-graphe connexe minimal couvrant T.

Supposons que l'algorithme s'arrête à l'itération $t+1$ et notons $K_i := (Y_i, L_i)$ l'arbre de Steiner et sa perte choisis à la i-ième itération ($i = 1, \ldots, t$). Soit c_0 la fonction coût $\bar{c}$ après l'étape ① et notons $c_i := c_{i-1}/K_i$ la fonction coût $\bar{c}$ après i itérations ($i = 1, \ldots, t$). Alors, d'après la proposition 20.11, l'algorithme calcule une solution de coût total au plus égal à $mst(c_t) + \sum_{i=1}^{t} c(L_i)$.

Soit Y^* un connecteur k-restreint optimum pour T ; soient $Y_1^*, \ldots, Y_{t^*}^*$ des arbres de Steiner complets k-restreints dont l'union est Y^* ; soient L_j^* une perte de Y_j^* et $K_j^* = (Y_j^*, L_j^*)$ ($j = 1, \ldots, t^*$) et soit $L^* := L_1^* \cup \cdots \cup L_{t^*}^*$. Nous écrivons c/K^* au lieu de $c/K_1^*/\cdots/K_{t^*}^*$. D'après la proposition 20.11, nous obtenons :

Lemme 20.13. *L'algorithme calcule un arbre de Steiner pour T de poids au plus égal à $mst(c_t) + \sum_{i=1}^{t} c(L_i)$. De plus, $c(E(Y^*)) = mst(c/K^*) + c(L^*)$.* □

Lemme 20.14. $mst(c_t) \le c(E(Y^*)) \le mst(c_0)$.

Preuve. $c(E(Y^*)) \le mst(c_0)$ est évident. Lorsque l'algorithme se termine, $mst(c_t) - mst(c_t/K_j^*) \le c(L_j^*)$ pour $j = 1, \ldots, t^*$. Ainsi, en utilisant le lemme 20.12,

$$\begin{aligned} mst(c_t) - mst(c/K^*) &\le mst(c_t) - mst(c_t/K^*) \\ &= \sum_{j=1}^{t^*} (mst(c_t/K_1^*/\cdots/K_{j-1}^*) - mst(c_t/K_1^*/\cdots/K_j^*)) \\ &\le \sum_{j=1}^{t^*} (mst(c_t) - mst(c_t/K_j^*)) \\ &\le \sum_{j=1}^{t^*} c(L_j^*), \end{aligned}$$

ce qui implique $mst(c_t) \le mst(c/K^*) + c(L^*)$. □

Lemme 20.15. $mst(c_t) + \sum_{i=1}^{t} c(L_i) \le c(E(Y^*))(1 + \frac{\ln 3}{2})$.

Preuve. Soit $i \in \{1, \ldots, t\}$. D'après le choix de L_i à l'itération i de l'algorithme,

$$\begin{aligned} \frac{mst(c_{i-1}) - mst(c_i)}{c(L_i)} &\ge \max_{j=1,\ldots,t^*} \frac{mst(c_{i-1}) - mst(c_{i-1}/K_j^*)}{c(L_j^*)} \\ &\ge \frac{\sum_{j=1}^{t^*} (mst(c_{i-1}) - mst(c_{i-1}/K_j^*))}{\sum_{j=1}^{t^*} c(L_j^*)} \\ &\ge \frac{\sum_{j=1}^{t^*} (mst(c_{i-1}/K_1^*/\cdots/K_{j-1}^*) - mst(c_{i-1}/K_1^*/\cdots/K_j^*))}{c(L^*)} \\ &= \frac{mst(c_{i-1}) - mst(c_{i-1}/K^*)}{c(L^*)} \\ &\ge \frac{mst(c_{i-1}) - mst(c/K^*)}{c(L^*)} \end{aligned}$$

(nous utilisons le lemme 20.12 pour la troisième inégalité et la monotonicité pour la dernière). De plus, le terme de gauche est au moins égal à 1. Ainsi

$$\begin{aligned} \sum_{i=1}^{t} c(L_i) &\le \sum_{i=1}^{t} (mst(c_{i-1}) - mst(c_i)) \frac{c(L^*)}{\max\{c(L^*), mst(c_{i-1}) - mst(c/K^*)\}} \\ &\le \int_{mst(c_t)}^{mst(c_0)} \frac{c(L^*)}{\max\{c(L^*), x - mst(c/K^*)\}} dx. \end{aligned}$$

Comme $c(E(Y^*)) = mst(c/K^*) + c(L^*)$ d'après le lemme 20.13, et comme $mst(c_t) \le c(E(Y^*)) \le mst(c_0)$ d'après le lemme 20.14, on calcule

$$\begin{aligned}\sum_{i=1}^{t} c(L_i) &\leq \int_{mst(c_t)}^{c(E(Y^*))} 1\,dx + \int_{c(L^*)}^{mst(c_0)-mst(c/K^*)} \frac{c(L^*)}{x}\,dx \\ &= c(E(Y^*)) - mst(c_t) + c(L^*)\ln\frac{mst(c_0)-mst(c/K^*)}{c(L^*)}.\end{aligned}$$

Comme $mst(c_0) \leq 2\,\mathrm{OPT}(G,c,T) \leq 2c(E(Y^*)) = c(E(Y^*)) + mst(c/K^*) + c(L^*)$, on obtient

$$mst(c_t) + \sum_{i=1}^{t} c(L_i) \leq c(E(Y^*))\left(1 + \frac{c(L^*)}{c(E(Y^*))}\ln\left(1 + \frac{c(E(Y^*))}{c(L^*)}\right)\right).$$

Comme $0 \leq c(L^*) \leq \frac{1}{2}c(E(Y^*))$ (d'après la proposition 20.10) et comme $\max\{x\ln(1+\frac{1}{x}) : 0 < x \leq \frac{1}{2}\}$ est atteint pour $x = \frac{1}{2}$ (puisque la dérivée $\ln(1+\frac{1}{x}) - \frac{1}{x+1}$ est toujours positive), on conclut que $mst(c_t) + \sum_{i=1}^{t} c(L_i) \leq c(E(Y^*))(1+\frac{\ln 3}{2})$. □

Cette preuve est essentiellement due à Gröpl *et al.* [2001]. Nous en concluons :

Théorème 20.16. (Robins et Zelikovsky [2005]) *L'algorithme de Robins-Zelikovsky a une garantie de performance égale à $\rho_k(1+\frac{\ln 3}{2})$ et s'exécute en temps polynomial pour chaque k fixé. Pour k suffisamment grand, la garantie de performance est inférieure à 1,55.*

Preuve. D'après le lemme 20.13, l'algorithme retourne un arbre de Steiner de coût au plus égal à $mst(c_t) + \sum_{i=1}^{t} c(L_i)$. D'après le lemme 20.15, ce coût est au plus égal à $\rho_k(1+\frac{\ln 3}{2})$. En choisissant $k = \min\{|V(G)|, 2^{2233}\}$ et en appliquant le théorème 20.9, on obtient un rapport de performance inférieur à $\rho_k(1+\frac{\ln 3}{2}) \leq \frac{2234}{2233}(1+\frac{\ln 3}{2}) < 1,55$.

Il y a au plus n^k sous-ensembles possibles S et pour chacun il existe au plus n^{k-2} choix pour les (au plus $k-2$) points de Steiner de degré au moins égal à 3 dans un arbre de Steiner optimum Y pour S. Alors, étant donné Y, il existe au plus $(2k-3)^{k-2}$ choix pour une perte (en incluant les arêtes de coût nul). Ainsi chaque itération prend un temps $O(n^{2k})$ (pour k fixé) et il y a au plus n^{2k-2} itérations. □

20.3 Conception de réseaux fiables

Avant de passer au PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE général, mentionnons deux autres cas particuliers. Si toutes les conditions de connexité r_{xy} sont égales à 0 ou 1, le problème est appelé le PROBLÈME DE L'ARBRE DE STEINER GÉNÉRALISÉ qui a pour cas particulier le PROBLÈME DE L'ARBRE DE STEINER. Le premier algorithme d'approximation pour le PROBLÈME DE L'ARBRE DE STEINER GÉNÉRALISÉ a été trouvé par Agrawal, Klein et Ravi [1995].

Un autre cas particulier intéressant est le problème de la recherche d'un sous-graphe k-arête-connexe de poids minimum (ici $r_{xy} = k$ pour tout x, y). Voir l'exercice 8 pour un algorithme d'approximation de facteur 2 combinatoire pour ce cas particulier et pour des références liées à ce problème.

Lorsque l'on étudie le PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE général, avec des conditions de connexité r_{xy} pour tout $x, y \in V(G)$, il est utile de considérer une fonction $f : 2^{V(G)} \to \mathbb{Z}_+$ définie par $f(\emptyset) := f(V(G)) := 0$ et $f(S) := \max_{x \in S,\, y \in V(G) \setminus S} r_{xy}$ pour $\emptyset \neq S \subset V(G)$. Alors notre problème peut être formulé comme le PL en nombres entiers suivant :

$$\begin{array}{lll} \min & \displaystyle\sum_{e \in E(G)} c(e) x_e & \\ \text{s.c.} & \displaystyle\sum_{e \in \delta(S)} x_e \;\geq\; f(S) & (S \subseteq V(G)) \\ & x_e \;\in\; \{0,1\} & (e \in E(G)). \end{array} \tag{20.1}$$

Nous ne traiterons pas ce PL en nombres entiers sous cette forme générale, mais nous utiliserons plutôt une propriété importante de f :

Définition 20.17. *Une fonction $f : 2^U \to \mathbb{Z}_+$ est dite* **propre** *si elle vérifie les trois conditions suivantes :*

- $f(S) = f(U \setminus S)$ *pour tout $S \subseteq U$;*
- $f(A \cup B) \leq \max\{f(A), f(B)\}$ *pour tout $A, B \subseteq U$ tels que $A \cap B = \emptyset$;*
- $f(\emptyset) = 0$.

Il est évident que la fonction f construite précédemment est propre. Les fonctions propres ont été introduites par Goemans et Williamson [1995] qui ont donné un algorithme d'approximation de facteur 2 pour les fonctions propres f telles que $f(S) \in \{0, 1\}$ pour tout S. Pour les fonctions propres f telles que $f(S) \in \{0, 2\}$ pour tout S, Klein et Ravi [1993] ont donné un algorithme d'approximation de facteur 3.

La propriété suivante des fonctions propres est essentielle :

Proposition 20.18. *Une fonction propre $f : 2^U \to \mathbb{Z}_+$ est* **faiblement supermodulaire***, c.-à-d. qu'au moins une des conditions suivantes est vérifiée pour tout $A, B \subseteq U$:*

- $f(A) + f(B) \;\leq\; f(A \cup B) + f(A \cap B)$,
- $f(A) + f(B) \;\leq\; f(A \setminus B) + f(B \setminus A)$.

Preuve. Par définition nous avons

$$
\begin{aligned}
f(A) &\leq \max\{f(A \setminus B), f(A \cap B)\}; && (20.2)\\
f(B) &\leq \max\{f(B \setminus A), f(A \cap B)\}; && (20.3)\\
f(A) &= f(U \setminus A) \leq \max\{f(B \setminus A), f(U \setminus (A \cup B))\} && (20.4)\\
&= \max\{f(B \setminus A), f(A \cup B)\};\\
f(B) &= f(U \setminus B) \leq \max\{f(A \setminus B), f(U \setminus (A \cup B))\} && (20.5)\\
&= \max\{f(A \setminus B), f(A \cup B)\}.
\end{aligned}
$$

On distingue maintenant quatre cas, suivant lequel des quatre nombres $f(A \setminus B), f(B \setminus A), f(A \cap B), f(A \cup B)$ est le plus petit. Si $f(A \setminus B)$ est le plus petit, on ajoute (20.2) et (20.5). Si $f(B \setminus A)$ est le plus petit, on ajoute (20.3) et (20.4). Si $f(A \cap B)$ est le plus petit, on ajoute (20.2) et (20.3). Si $f(A \cup B)$ est le plus petit, on ajoute (20.4) et (20.5). □

Dans le reste de ce paragraphe, nous montrons comment résoudre la relaxation linéaire de (20.1) :

$$
\begin{aligned}
\min \quad & \sum_{e \in E(G)} c(e) x_e && && \\
\text{s.c.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S) && (S \subseteq V(G)) && (20.6)\\
& x_e \geq 0 && (e \in E(G)) && \\
& x_e \leq 1 && (e \in E(G)). &&
\end{aligned}
$$

On ne sait pas comment résoudre ce PL en temps polynomial pour des fonctions quelconques f, pas même pour des fonctions faiblement super-modulaires quelconques. Par conséquent nous nous restreignons au cas où f est une fonction propre. D'après le théorème 4.21, il suffit de résoudre le PROBLÈME DE SÉPARATION. Nous utilisons un arbre de Gomory-Hu :

Lemme 20.19. *Soit G un graphe non orienté avec des capacités $u \in \mathbb{R}_+^{E(G)}$ et soit $f : 2^{V(G)} \to \mathbb{Z}_+$ une fonction propre. Soit H un arbre de Gomory-Hu pour (G, u). Alors pour tout $\emptyset \neq S \subset V(G)$, on a :*

(a) $\sum_{e' \in \delta_G(S)} u(e') \geq \max_{e \in \delta_H(S)} \sum_{e' \in \delta_G(C_e)} u(e')$;

(b) $f(S) \leq \max_{e \in \delta_H(S)} f(C_e)$;

où C_e et $V(H) \setminus C_e$ sont les deux composantes connexes de $H - e$.

Preuve. (a) : par définition de l'arbre de Gomory-Hu, $\delta_G(C_e)$ est une coupe séparant x et y de capacité minimum pour chaque $e = (x, y) \in E(H)$, et pour $(x, y) \in \delta_H(S)$ le terme de gauche dans (a) est la capacité d'une certaine coupe séparant x et y.

Afin de prouver (b), considérons les composantes connexes $X_1, \ldots, X_l$ de $H - S$. Puisque $H[X_i]$ est connexe et que H est un arbre, nous avons pour chaque $i \in \{1, \ldots, l\}$:

$$V(H) \setminus X_i = \bigcup_{e \in \delta_H(X_i)} C_e$$

(si nécessaire, remplacer C_e par $V(H) \setminus C_e$). Comme f est propre, nous avons

$$f(X_i) \;=\; f(V(G)\backslash X_i) \;=\; f(V(H)\backslash X_i) \;=\; f\left(\bigcup_{e\in\delta_H(X_i)} C_e\right) \;\leq\; \max_{e\in\delta_H(X_i)} f(C_e).$$

Comme $\delta_H(X_i) \subseteq \delta_H(S)$, nous concluons que

$$f(S) \;=\; f(V(G)\setminus S) \;=\; f\left(\bigcup_{i=1}^{l} X_i\right) \;\leq\; \max_{i\in\{1,\ldots,l\}} f(X_i) \;\leq\; \max_{e\in\delta_H(S)} f(C_e).$$

□

Montrons maintenant comment résoudre le PROBLÈME DE SÉPARATION pour (20.6) en considérant les coupes fondamentales d'un arbre de Gomory-Hu. Remarquons que stocker la fonction propre f explicitement nécessiterait un espace mémoire exponentiel. Nous supposons donc que f est donnée par un oracle.

Théorème 20.20. *Soit G un graphe non orienté, $x \in \mathbb{R}_+^{E(G)}$ et soit $f : 2^{V(G)} \to \mathbb{Z}_+$ une fonction propre (donnée par un oracle). On peut trouver un ensemble $S \subseteq V(G)$ tel que $\sum_{e\in\delta_G(S)} x_e < f(S)$ ou conclure qu'un tel ensemble n'existe pas en temps $O\left(n^4 + n\theta\right)$. Ici $n = |V(G)|$ et θ est le temps requis par l'oracle pour f.*

Preuve. Nous calculons d'abord un arbre de Gomory-Hu H pour G, où les capacités sont données par x. H peut être calculé en temps $O(n^4)$ d'après le théorème 8.35.

D'après le lemme 20.19(b), nous avons que pour chaque $\emptyset \neq S \subset V(G)$ il existe un $e \in \delta_H(S)$ tel que $f(S) \leq f(C_e)$. D'après le lemme 20.19(a), nous obtenons $f(S) - \sum_{e\in\delta_G(S)} x_e \leq f(C_e) - \sum_{e\in\delta_G(C_e)} x_e$. Nous en concluons

$$\max_{\emptyset\neq S\subset V(G)} \left(f(S) - \sum_{e\in\delta_G(S)} x_e\right) \;=\; \max_{e\in E(H)} \left(f(C_e) - \sum_{e\in\delta_G(C_e)} x_e\right). \quad (20.7)$$

Ainsi le PROBLÈME DE SÉPARATION pour (20.6) peut être résolu en vérifiant seulement $n-1$ coupes. □

Il est intéressant de comparer (20.7) au théorème 12.17.

Contrairement à la relaxation linéaire (20.6), on ne peut espérer trouver une solution entière optimale en temps polynomial : d'après le théorème 20.2, cela impliquerait $P = NP$. Nous considérons donc des algorithmes d'approximation pour (20.1).

Dans le paragraphe suivant, nous décrivons un algorithme d'approximation primal-dual qui ajoute consécutivement des arêtes aux coupes les plus violées. Cet algorithme combinatoire fonctionne bien si la condition de connexité maximale $k := \max_{S\subseteq V(G)} f(S)$ n'est pas trop grande. En particulier, il s'agit d'un algorithme d'approximation de facteur 2 dans le cas où $k = 1$, qui inclut le PROBLÈME

DE L'ARBRE DE STEINER GÉNÉRALISÉ. Au paragraphe 20.5, nous décrivons un algorithme d'approximation de facteur 2 pour le cas général. Cependant, cet algorithme a l'inconvénient d'utiliser la solution précédente de la relaxation linéaire (20.6), qui a un temps de calcul polynomial, mais n'est pas assez efficace en pratique.

20.4 Un algorithme d'approximation primal-dual

L'algorithme que nous décrivons dans ce paragraphe a été présenté dans les articles de Williamson *et al.* [1995], Gabow, Goemans et Williamson [1998] et Goemans *et al.* [1994], dans cet ordre.

Étant donné un graphe non orienté G avec des poids $c : E(G) \to \mathbb{R}_+$ et une fonction propre f, nous recherchons un ensemble d'arêtes F dont le vecteur d'incidence vérifie (20.1).

L'algorithme procède en $k := \max_{S \subseteq V(G)} f(S)$ phases. Puisque f est propre, nous avons $k = \max_{v \in V(G)} f(\{v\})$, donc k peut être calculé facilement. À l'étape p ($1 \leq p \leq k$), on considère la fonction propre f_p définie par $f_p(S) := \max\{f(S) + p - k, 0\}$. Nous garantirons qu'après la phase p l'ensemble d'arêtes courant F (ou, plus précisément, son vecteur d'incidence) vérifie (20.1) par rapport à f_p. Commençons par quelques définitions.

Définition 20.21. *Étant donné une fonction propre g, $F \subseteq E(G)$ et $X \subseteq V(G)$, on dit que X est* **violé** *respectivement à (g, F) si $|\delta_F(X)| < g(X)$. Les ensembles violés minimum par rapport à (g, F) sont les ensembles* **actifs** *respectivement à (g, F). $F \subseteq E(G)$* **satisfait** *g si aucun ensemble n'est violé respectivement à (g, F). On dit que F* **satisfait presque** *g si $|\delta_F(X)| \geq g(X) - 1$ pour tout $X \subseteq V(G)$.*

Au cours de l'exécution de l'algorithme, la fonction courante f_p sera presque satisfaite par l'ensemble courant F. Les ensembles actifs joueront un rôle central. L'observation suivante est fondamentale :

Lemme 20.22. *Étant donné une fonction propre g, un ensemble $F \subseteq E(G)$ satisfaisant presque g et deux ensembles violés A et B. Alors soit $A \setminus B$ et $B \setminus A$ sont tous deux violés, soit $A \cap B$ et $A \cup B$ sont tous deux violés. En particulier, les ensembles actifs respectivement à (g, F) sont deux à deux disjoints.*

Preuve. Directement à partir de la proposition 20.18 et des parties (c) et (d) du lemme 2.1. □

Ce lemme montre en particulier qu'il peut exister au plus $n = |V(G)|$ ensembles actifs. Nous montrons maintenant comment calculer les ensembles actifs. De manière similaire à la preuve du théorème 20.20, nous utilisons un arbre de Gomory-Hu.

Théorème 20.23. (Gabow, Goemans et Williamson [1998]) *Soit une fonction propre g (donnée par un oracle) et un ensemble $F \subseteq E(G)$ satisfaisant presque g ; alors les ensembles actifs respectivement à (g, F) peuvent être calculés en temps $O\left(n^4 + n^2\theta\right)$. Ici $n = |V(G)|$ et θ est le temps requis par l'oracle pour g.*

Preuve. Nous calculons d'abord un arbre de Gomory-Hu H pour $(V(G), F)$ (et des capacités unitaires). H peut être calculé en temps $O(n^4)$ d'après le théorème 8.35. D'après le lemme 20.19, on a pour chaque $\emptyset \neq S \subset V(G)$:

$$|\delta_F(S)| \;\geq\; \max_{e \in \delta_H(S)} |\delta_F(C_e)| \tag{20.8}$$

et

$$g(S) \;\leq\; \max_{e \in \delta_H(S)} g(C_e), \tag{20.9}$$

où C_e et $V(H) \setminus C_e$ sont les deux composantes connexes de $H - e$.

Soit A un ensemble actif. D'après (20.9), il existe une arête $e = (s, t) \in \delta_H(A)$ telle que $g(A) \leq g(C_e)$. D'après (20.8), $|\delta_F(A)| \geq |\delta_F(C_e)|$. On a donc

$$1 \;=\; g(A) - |\delta_F(A)| \;\leq\; g(C_e) - |\delta_F(C_e)| \;\leq\; 1,$$

puisque F satisfait presque g. On doit avoir égalité partout, en particulier $|\delta_F(A)| = |\delta_F(C_e)|$. Donc $\delta_F(A)$ est une coupe minimum séparant s et t dans $(V(G), F)$. Supposons sans perte de généralité que A contienne t mais pas s.

Soit G' le graphe orienté obtenu en remplaçant chaque arête par deux arcs de direction opposée. Considérons un flot maximum de s à t, f dans G' et dans le graphe résiduel G'_f. Formons un graphe orienté acyclique G'' à partir de G'_f obtenu en contractant l'ensemble S des sommets connectés à s en un sommet v_S, puis en contractant l'ensemble T des sommets depuis lesquels on peut atteindre t en un sommet v_T et en contractant enfin chaque composante fortement connexe X de $G'_f - (S \cup T)$ en un sommet v_X. Il existe une bijection entre les coupes séparant t de s minimum dans G' et les coupes orientées séparant v_S de v_T dans G'' (exercice 5 du chapitre 8 ; c'est une conséquence immédiate du théorème flot-max/coupe-min 8.6 et du lemme 8.3). En particulier, A est l'union des ensembles X tels que $v_X \in V(G'')$. Puisque $g(A) > |\delta_F(A)| = |\delta^-_{G'}(A)| = \text{valeur}\,(f)$ et que g est propre, il existe un sommet $v_X \in V(G'')$ tel que $X \subseteq A$ et $g(X) > \text{valeur}\,(f)$.

Nous montrons maintenant comment trouver A. Si $g(T) > \text{valeur}\,(f)$, alors posons $Z := T$, sinon soit v_Z un sommet de G'' tel que $g(Z) > \text{valeur}\,(f)$ et $g(Y) \leq \text{valeur}\,(f)$ pour tous les sommets $v_Y \in V(G'') \setminus \{v_Z\}$ desquels on peut atteindre v_Z. Soit

$$B \;:=\; T \cup \bigcup \{Y : v_Z \text{ est connecté à } v_Y \text{ dans } G''\}.$$

Puisque

$$\begin{aligned} \text{valeur}\,(f) \;<\; g(Z) \;=\; g(V(G) \setminus Z) \;&\leq\; \max\{g(V(G) \setminus B), g(B \setminus Z)\} \\ &=\; \max\{g(B), g(B \setminus Z)\} \end{aligned}$$

et que

$$g(B \setminus Z) \;\leq\; \max\{g(Y) : v_Y \in V(G'') \setminus \{v_Z\},\, Y \subseteq B\} \;\leq\; \text{valeur}\,(f)$$

on a $g(B) > \text{valeur}\,(f) = |\delta^-_{G'}(B)| = |\delta_F(B)|$, donc B est violé respectivement à (g, F). Comme B n'est pas un sous-ensemble propre de A (puisque A est actif) et puisque A et B contiennent tous deux T, nous concluons du lemme 20.22 que $A \subseteq B$. Mais alors $Z = X$, puisque v_Z est le seul sommet tel que $Z \subseteq B$ et $g(Z) > \text{valeur}\,(f)$ et A contient tous les ensembles Y pour lesquels on peut atteindre v_Z à partir de v_Y (puisque $\delta^-_{G'_f}(A) = \emptyset$). Ainsi $A = B$.

Pour un couple (s, t) donné, un ensemble B comme ci-dessus (s'il existe) peut être trouvé en temps linéaire en construisant G'' (en utilisant l'ALGORITHME DES COMPOSANTES FORTEMENT CONNEXES) et en trouvant ensuite un ordre topologique de G'' (voir théorème 2.20), commençant par v_T. Nous répétons la procédure précédente pour tous les couples ordonnés (s, t) tels que $(s, t) \in E(H)$.

De cette façon, nous obtenons une liste d'au plus $2n - 2$ candidats pour les ensembles actifs. Le temps de calcul est évidemment dominé par le temps nécessaire pour trouver $O(n)$ fois un flot maximum dans G' et faire appel $O(n^2)$ fois à l'oracle pour g. Enfin on peut éliminer, parmi les ensembles candidats, les ensembles violés qui ne sont pas minimaux en temps $O(n^2)$. □

Le temps de calcul peut être amélioré si $\max_{S \subseteq V(G)} g(S)$ est petit (voir exercice 10). Nous passons maintenant à la description de l'algorithme.

ALGORITHME PRIMAL-DUAL POUR LA CONCEPTION DE RÉSEAU	
Input	Un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$ et un oracle pour une fonction propre $f : 2^{V(G)} \to \mathbb{Z}_+$.
Output	Un ensemble $F \subseteq E(G)$ satisfaisant f.

① **If** $E(G)$ ne satisfait pas f, **then stop** (le problème est non réalisable).

② Poser $F := \emptyset$, $k := \max_{v \in V(G)} f(\{v\})$ et $p := 1$.

③ Poser $i := 0$.
Poser $\pi(v) := 0$ pour tout $v \in V(G)$.
Soit $\mathcal{A}$ la famille des ensembles actifs respectivement à (F, f_p), où f_p est défini par $f_p(S) := \max\{f(S) + p - k, 0\}$ pour tout $S \subseteq V(G)$.

④ **While** $\mathcal{A} \neq \emptyset$ **do** :
Poser $i := i + 1$.
Poser $\epsilon := \min\left\{ \dfrac{c(e) - \pi(v) - \pi(w)}{|\{A \in \mathcal{A} :\; e \in \delta_G(A)\}|} : e = (v, w) \in \bigcup_{A \in \mathcal{A}} \delta_G(A) \setminus F \right\}$,
et soit e_i une arête pour laquelle le minimum est atteint.
Augmenter $\pi(v)$ par ϵ pour tout $v \in \bigcup_{A \in \mathcal{A}} A$.
Poser $F := F \cup \{e_i\}$.
Mettre à jour $\mathcal{A}$.

⑤ **For** $j := i$ **down to** 1 **do** :
If $F \setminus \{e_j\}$ satisfait f_p **then** poser $F := F \setminus \{e_j\}$.

⑥ **If** $p = k$ **then stop**, **else** poser $p := p + 1$ et **go to** ③.

La vérification de réalisabilité à l'étape ① peut être faite en temps $O\left(n^4 + n\theta\right)$ d'après le théorème 20.20. Avant de présenter comment implémenter les étapes ③ et ④, montrons que l'ensemble F renvoyé par l'algorithme est effectivement réalisable relativement à f. Notons F_p l'ensemble F à la fin de la phase p (et $F_0 := \emptyset$).

Lemme 20.24. *À chaque étape de la phase p, l'ensemble F satisfait presque f_p et $F \setminus F_{p-1}$ est une forêt. À la fin de la phase p, F_p satisfait f_p.*

Preuve. Comme $f_1(S) = \max\{0, f(S) + 1 - k\} \leq \max\{0, \max_{v \in S} f(\{v\}) + 1 - k\} \leq 1$ (puisque f est propre), l'ensemble vide satisfait presque f_1.

Après l'étape ④, il n'existe plus d'ensembles actifs, donc F satisfait f_p. À l'étape ⑤, cette propriété est explicitement conservée. Donc chaque F_p satisfait f_p et ainsi satisfait presque f_{p+1} ($p = 0, \ldots, k-1$). Pour voir que $F \setminus F_{p-1}$ est une forêt, observons que chaque arête ajoutée à F appartient à $\delta(A)$ pour un certain ensemble actif A et doit être la première arête de $\delta(A)$ ajoutée à F dans cette phase (puisque $|\delta_{F_{p-1}}(A)| = f_{p-1}(A)$). Ainsi aucune arête ne crée un cycle dans $F \setminus F_{p-1}$. □

Le théorème 20.23 peut donc être appliqué pour déterminer $\mathcal{A}$. Le nombre d'itérations au cours de chaque phase est au plus égal à $n-1$. Concernant l'implémentation, il ne nous reste plus qu'à voir comment déterminer ϵ et e_i à l'étape ④.

Lemme 20.25. *On peut déterminer ϵ et e_i à l'étape ④ de l'algorithme en temps $O(mn)$ pour chaque phase.*

Preuve. À chaque itération d'une phase, nous suivons la procédure suivante. Tout d'abord nous affectons un nombre à chaque sommet selon l'ensemble actif auquel il appartient (ou zéro s'il n'appartient à aucun ensemble actif). Cela peut être fait en temps $O(n)$ (remarquons que les ensembles actifs sont deux à deux disjoints d'après le lemme 20.22). Pour chaque arête e, le nombre d'ensembles actifs contenant exactement une extrémité de e peut maintenant être déterminé en temps $O(1)$. Donc ϵ et e_i peuvent être déterminés en temps $O(m)$. Il y a au plus $n-1$ itérations par phase, donc la borne est prouvée. □

Remarquons que Gabow, Goemans et Williamson [1998] ont amélioré cette borne jusqu'à $O\left(n^2\sqrt{\log\log n}\right)$ à l'aide d'une implémentation sophistiquée.

Théorème 20.26. (Goemans *et al.* [1994]) *L'*ALGORITHME PRIMAL-DUAL POUR LA CONCEPTION DE RÉSEAU *retourne un ensemble F satisfaisant f en temps $O\left(kn^5 + kn^3\theta\right)$, où $k = \max_{S \subseteq V(G)} f(S)$, $n = |V(G)|$ et θ est le temps requis par l'oracle pour f.*

Preuve. La réalisabilité de F est garantie par le lemme 20.24 puisque $f_k = f$.

Un oracle pour chaque f_p utilise bien entendu l'oracle pour f et requiert donc un temps $\theta + O(1)$. Le calcul des ensembles actifs peut être réalisé en temps $O\left(n^4 + n^2\theta\right)$ (théorème 20.23) et ce calcul est fait $O(nk)$ fois. Déterminer ϵ et e_i peut être réalisé en temps $O(n^3)$ pour chaque phase (lemme 20.25). Tout le reste peut facilement être fait en temps $O(kn^2)$. □

L'exercice 10 montre comment améliorer ce temps jusqu'à $O\left(k^3n^3 + kn^3\theta\right)$. Il peut être amélioré jusqu'à $O\left(k^2n^3 + kn^2\theta\right)$ en utilisant une étape de mise à jour différente (l'étape ⑤ de l'algorithme) et une implémentation plus sophistiquée (Gabow, Goemans et Williamson [1998]). Pour k fixé et $\theta = O(n)$, cela signifie que nous avons un algorithme en $O\left(n^3\right)$. Pour le cas particulier du PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE (f est déterminé par les conditions de connexité r_{xy}) le temps de calcul peut être amélioré jusqu'à $O\left(k^2n^2\sqrt{\log\log n}\right)$.

Nous analysons maintenant la garantie de performance de l'algorithme et justifions qu'il s'agit d'un algorithme primal-dual. Le dual de (20.6) est :

$$\begin{array}{lrcll} \max & \displaystyle\sum_{S\subseteq V(G)} f(S)\, y_S & - & \displaystyle\sum_{e\in E(G)} z_e & \\ \text{s.c.} & \displaystyle\sum_{S:e\in\delta(S)} y_S & \leq & c(e) + z_e & (e \in E(G)) \\ & y_S & \geq & 0 & (S \subseteq V(G)) \\ & z_e & \geq & 0 & (e \in E(G)). \end{array} \tag{20.10}$$

Ce PL dual est essentiel pour l'analyse de l'algorithme.

Nous montrons comment l'algorithme construit implicitement à chaque phase p une solution duale réalisable $y^{(p)}$. Partant de $y^{(p)} = 0$, on augmente, pour chaque $A \in \mathcal{A}$, $y_A^{(p)}$ de ϵ à chaque itération de cette phase. De plus nous posons

$$z_e^{(p)} := \begin{cases} \displaystyle\sum_{S:\ e\in\delta(S)} y_S^{(p)} & \text{si } e \in F_{p-1} \\ 0 & \text{sinon} \end{cases}.$$

Il n'y a pas lieu de construire cette solution duale explicitement dans l'algorithme. Les variables $\pi(v) = \sum_{S:v\in S} y_S$ $(v \in V(G))$ contiennent toute l'information nécessaire.

Lemme 20.27. (Williamson *et al.* [1995]) *Pour chaque p, $(y^{(p)}, z^{(p)})$ est une solution réalisable de (20.10).*

Preuve. Les contraintes de non négativité sont évidemment vérifiées. Les contraintes pour $e \in F_{p-1}$ sont vérifiées par définition de $z_e^{(p)}$.

De plus, d'après l'étape ④ de l'algorithme, nous avons

$$\sum_{S:e\in\delta(S)} y_S^{(p)} \;\leq\; c(e) \quad \text{pour chaque } e \in E(G)\setminus F_{p-1},$$

puisque e est ajouté à F lorsque l'on atteint égalité et après que les ensembles S tels que $e \in \delta(S)$ ne sont plus violés respectivement à (F, f_p) (rappelons que $F\setminus\{e\} \supseteq F_{p-1}$ satisfait presque f_p d'après le lemme 20.24). □

Notons $\mathrm{OPT}(G, c, f)$ la valeur optimale du PL en nombres entiers (20.1). Nous montrons ensuite :

Lemme 20.28. (Goemans *et al.* [1994]) *Pour chaque $p \in \{1,\dots,k\}$, on a*

$$\sum_{S\subseteq V(G)} y_S^{(p)} \;\leq\; \frac{1}{k-p+1}\,\mathrm{OPT}(G,c,f).$$

Preuve. $\mathrm{OPT}(G, c, f)$ est supérieur ou égal à la valeur optimale de la relaxation linéaire (20.6) et celle-ci est bornée inférieurement par la valeur de la fonction objectif atteinte par une solution réalisable du dual (d'après le théorème de dualité 3.20). Puisque $(y^{(p)}, z^{(p)})$ est réalisable pour le PL dual (20.10) d'après le lemme 20.27, on conclut que

$$\mathrm{OPT}(G,c,f) \;\geq\; \sum_{S\subseteq V(G)} f(S)\, y_S^{(p)} - \sum_{e\in E(G)} z_e^{(p)}.$$

Observons maintenant que, pour chaque $S \subseteq V(G)$, y_S peut seulement devenir positif si S est violé respectivement à (f_p, F_{p-1}). Nous pouvons donc conclure que

$$y_S^{(p)} > 0 \;\Rightarrow\; |\delta_{F_{p-1}}(S)| \leq f(S) + p - k - 1.$$

Nous obtenons ainsi :

$$\begin{aligned}
\mathrm{OPT}(G,c,f) \;&\geq\; \sum_{S\subseteq V(G)} f(S)\, y_S^{(p)} - \sum_{e\in E(G)} z_e^{(p)} \\
&=\; \sum_{S\subseteq V(G)} f(S)\, y_S^{(p)} - \sum_{e\in F_{p-1}} \left(\sum_{S:e\in\delta(S)} y_S^{(p)} \right) \\
&=\; \sum_{S\subseteq V(G)} f(S)\, y_S^{(p)} - \sum_{S\subseteq V(G)} |\delta_{F_{p-1}}(S)|\, y_S^{(p)} \\
&=\; \sum_{S\subseteq V(G)} \left(f(S) - |\delta_{F_{p-1}}(S)|\right) y_S^{(p)} \\
&\geq\; \sum_{S\subseteq V(G)} (k-p+1)\, y_S^{(p)}.
\end{aligned}$$

□

Lemme 20.29. (Williamson *et al.* [1995]) *À chaque itération d'une phase p, nous avons*

$$\sum_{A\in\mathcal{A}} |\delta_{F_p\setminus F_{p-1}}(A)| \;\leq\; 2\,|\mathcal{A}|.$$

Preuve. Nous considérons une certaine itération de la phase p, que nous appelons l'itération courante. Notons $\mathcal{A}$ la famille des ensembles actifs au début de cette itération, et définissons

$$H := (F_p \setminus F_{p-1}) \cap \bigcup_{A \in \mathcal{A}} \delta(A).$$

Remarquons que toutes les arêtes de H doivent avoir été ajoutées pendant ou après l'itération courante.

Soit $e \in H$. $F_p \setminus \{e\}$ ne satisfait pas f_p, car sinon e aurait été supprimée dans l'étape de mise à jour ⑤ de la phase p. Soit donc X_e un ensemble violé respectivement à $(f_p, F_p \setminus \{e\})$ de cardinalité minimum. Comme $F_p \setminus \{e\} \supseteq F_{p-1}$ satisfait presque f_p, nous avons $\delta_{F_p \setminus F_{p-1}}(X_e) = \{e\}$.

Nous affirmons que la famille $\mathcal{X} := \{X_e : e \in H\}$ est laminaire. Pour le prouver, supposons qu'il existe deux arêtes $e, e' \in H$ (disons que e a été ajouté avant e') pour lesquelles $X_e \setminus X_{e'}$, $X_{e'} \setminus X_e$ et $X_e \cap X_{e'}$ sont tous non vides. Comme X_e et $X_{e'}$ sont violés au début de l'itération courante, soit $X_e \cup X_{e'}$ et $X_e \cap X_{e'}$ sont tous deux violés, soit $X_e \setminus X_{e'}$ et $X_{e'} \setminus X_e$ sont tous deux violés (d'après le lemme 20.22). Dans le premier cas, nous avons

$$\begin{aligned} 1+1 &\leq |\delta_{F_p \setminus F_{p-1}}(X_e \cup X_{e'})| + |\delta_{F_p \setminus F_{p-1}}(X_e \cap X_{e'})| \\ &\leq |\delta_{F_p \setminus F_{p-1}}(X_e)| + |\delta_{F_p \setminus F_{p-1}}(X_{e'})| = 1+1 \end{aligned}$$

par la sous-modularité de $|\delta_{F_p \setminus F_{p-1}}|$ (lemme 2.1(c)).
Nous en concluons que $|\delta_{F_p \setminus F_{p-1}}(X_e \cup X_{e'})| = |\delta_{F_p \setminus F_{p-1}}(X_e \cap X_{e'})| = 1$, ce qui contredit le choix de X_e ou de $X_{e'}$, car on aurait pu choisir $X_e \cap X_{e'}$ à la place. Dans le deuxième cas, d'après le lemme 2.1(d), $|\delta_{F_p \setminus F_{p-1}}(X_e \setminus X_{e'})| = |\delta_{F_p \setminus F_{p-1}}(X_{e'} \setminus X_e)| = 1$ et l'ensemble le plus petit parmi $X_e \setminus X_{e'}$ et $X_{e'} \setminus X_e$ contredit le choix de X_e ou $X_{e'}$.

Considérons maintenant une représentation par arbre (T, φ) de $\mathcal{X}$, où T est une arborescence (voir proposition 2.14). Pour chaque $e \in H$, X_e est violé au début de l'itération courante, car e n'a pas encore été ajouté à ce moment-là. Donc d'après le lemme 20.22, on a $A \subseteq X_e$ ou $A \cap X_e = \emptyset$ pour tout $A \in \mathcal{A}$. Ainsi $\{\varphi(a) : a \in A\}$ contient un seul élément, noté $\varphi(A)$, pour chaque $A \in \mathcal{A}$. On dit qu'un sommet $v \in V(T)$ est **occupé** si $v = \varphi(A)$ pour un certain $A \in \mathcal{A}$.

Nous affirmons que tous les sommets de T de degré sortant égal à 0 sont occupés. Pour un tel sommet v, $\varphi^{-1}(v)$ est un élément minimal de $\mathcal{X}$. Un élément minimal de $\mathcal{X}$ est violé au début de l'itération courante, il contient donc un ensemble actif et doit ainsi être occupé. Donc le degré sortant moyen des sommets occupés est inférieur à un.

Observons qu'il existe une bijection entre H, $\mathcal{X}$ et $E(T)$ (voir figure 20.3 ; (a) représente H, les éléments de $\mathcal{A}$ (les carrés) et ceux de $\mathcal{X}$ (les cercles) ; (b) représente T). Nous concluons que pour chaque $v \in V(T)$

$$|\delta_T(v)| = |\delta_H(\{x \in V(G) : \varphi(x) = v\})| \geq \sum_{A \in \mathcal{A} : \varphi(A) = v} |\delta_{F_p \setminus F_{p-1}}(A)|.$$

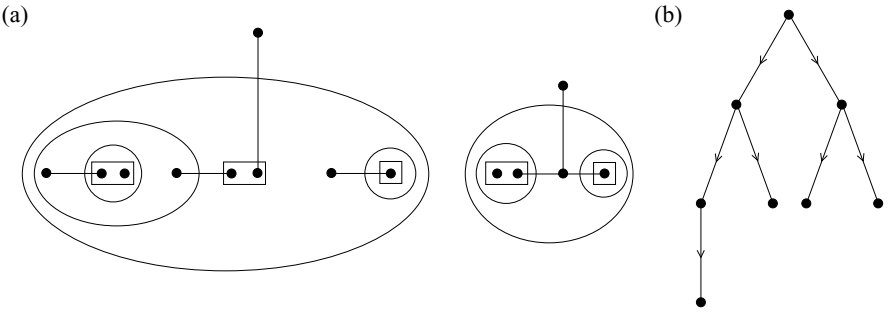


Figure 20.3.

En sommant sur tous les sommets occupés, on obtient :

$$\begin{aligned}\sum_{A\in\mathcal{A}} |\delta_{F_p\setminus F_{p-1}}(A)| &\leq \sum_{v\in V(T) \text{ occupé}} |\delta_T(v)| \\ &< 2\,|\{v \in V(T) : v \text{ occupé}\}| \\ &\leq 2\,|\mathcal{A}|.\end{aligned}$$

□

La preuve du lemme suivant montre le rôle des conditions des écarts complémentaires :

Lemme 20.30. (Williamson *et al.* [1995]) *Pour chaque $p \in \{1,\dots,k\}$, on a*

$$\sum_{e\in F_p\setminus F_{p-1}} c(e) \;\leq\; 2 \sum_{S\subseteq V(G)} y_S^{(p)}.$$

Preuve. À chaque phase p, l'algorithme maintient les conditions des écarts complémentaires du primal

$$e \in F \setminus F_{p-1} \;\Rightarrow\; \sum_{S:e\in\delta(S)} y_S^{(p)} = c(e).$$

Nous avons donc

$$\sum_{e\in F_p\setminus F_{p-1}} c(e) \;=\; \sum_{e\in F_p\setminus F_{p-1}} \left(\sum_{S:e\in\delta(S)} y_S^{(p)} \right) \;=\; \sum_{S\subseteq V(G)} y_S^{(p)}\, |\delta_{F_p\setminus F_{p-1}}(S)|.$$

Il reste ainsi à montrer que

$$\sum_{S\subseteq V(G)} y_S^{(p)}\, |\delta_{F_p\setminus F_{p-1}}(S)| \;\leq\; 2 \sum_{S\subseteq V(G)} y_S^{(p)}. \tag{20.11}$$

Au début de la phase p, nous avons $y^{(p)} = 0$, donc (20.11) est vérifié. À chaque itération, le terme de gauche augmente de $\sum_{A\in\mathcal{A}} \epsilon |\delta_{F_p\setminus F_{p-1}}(A)|$, tandis que le

terme de droite augmente de $2\epsilon|\mathcal{A}|$. Donc le lemme 20.29 montre que (20.11) n'est pas violé. □

Dans (20.11), les conditions des écarts complémentaires du dual

$$y_S^{(p)} > 0 \;\Rightarrow\; |\delta_{F_p}(S)| = f_p(S)$$

apparaissent. $|\delta_{F_p}(S)| \geq f_p(S)$ est toujours vérifié, tandis que (20.11) signifie que $|\delta_{F_p}(S)| \leq 2f_p(S)$ est vérifié en moyenne. Comme nous le verrons, cela implique un rapport de performance égal à 2 dans le cas où $k = 1$.

Théorème 20.31. (Goemans *et al.* [1994]) *L'*ALGORITHME PRIMAL-DUAL POUR LA CONCEPTION DE RÉSEAU *retourne un ensemble F qui satisfait f et dont le poids est au plus égal à $2H(k)\,\mathrm{OPT}(G, c, f)$ en temps $O\left(kn^5 + kn^3\theta\right)$, où $n = |V(G)|$, $k = \max_{S\subseteq V(G)} f(S)$, $H(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$ et θ est le temps requis par l'oracle pour f.*

Preuve. L'exactitude et le temps de calcul ont été prouvés au théorème 20.26. Le poids de F est

$$\begin{aligned}
\sum_{e\in F} c(e) &= \sum_{p=1}^{k} \left(\sum_{e\in F_p\setminus F_{p-1}} c(e) \right) \\
&\leq \sum_{p=1}^{k} \left(2 \sum_{S\subseteq V(G)} y_S^{(p)} \right) \\
&\leq 2\sum_{p=1}^{k} \frac{1}{k-p+1}\,\mathrm{OPT}(G, c, f) \\
&= 2H(k)\,\mathrm{OPT}(G, c, f)
\end{aligned}$$

d'après le lemme 20.30 et le lemme 20.28. □

L'algorithme d'approximation primal-dual présenté dans ce paragraphe a été décrit dans un cadre plus général par Bertsimas et Teo [1995]. Un problème connexe, mais apparemment plus difficile, apparaît si on considère la sommet-connexité au lieu de l'arête-connexité (on recherche un sous-graphe contenant au moins un nombre spécifique r_{ij} de chaînes de i à j sommet-disjointes pour chaque i et j). Voir les remarques à la fin du paragraphe suivant.

20.5 Algorithme de Jain

Dans ce paragraphe, nous présentons l'algorithme d'approximation de facteur 2 de Jain [2001] pour le PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE. Bien qu'il ait une garantie de performance meilleure que l'ALGORITHME PRIMAL-DUAL

POUR LA CONCEPTION DE RÉSEAU, son utilité pratique est moins bonne puisqu'il est fondé sur l'équivalence entre optimisation et séparation (voir paragraphe 4.6).

L'algorithme commence par résoudre la relaxation linéaire (20.6). Nous pouvons considérer des capacités entières $u : E(G) \to \mathbb{N}$ sur les arêtes puisque cela n'ajoute pas de difficulté, c.-à-d. que nous pouvons sélectionner certaines arêtes plus d'une fois :

$$\begin{array}{lll} \min & \displaystyle\sum_{e\in E(G)} c(e)x_e & \\ \text{s.c.} & \displaystyle\sum_{e\in\delta(S)} x_e \geq f(S) & (S \subseteq V(G)) \\ & x_e \geq 0 & (e \in E(G)) \\ & x_e \leq u(e) & (e \in E(G)). \end{array} \tag{20.12}$$

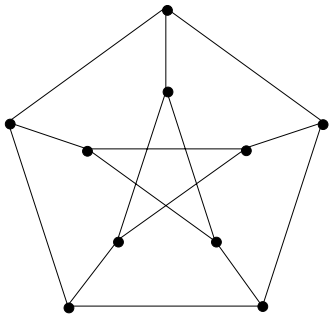


Figure 20.4.

Bien entendu nous recherchons par la suite une solution entière. En résolvant une relaxation linéaire d'un PL en nombres entiers et en arrondissant supérieurement la solution, on obtient un algorithme d'approximation de facteur 2 si la relaxation linéaire a toujours une solution optimale demi-entière (voir l'exercice 6 du chapitre 16 pour un exemple).

Cependant (20.12) ne vérifie pas cette propriété. Pour voir cela, considérons le graphe de Petersen (figure 20.4) avec $u(e) = c(e) = 1$ pour toute arête e et $f(S) = 1$ pour tout $\emptyset \neq S \subset V(G)$. Alors la valeur optimale du PL (20.12) est égale à 5 ($x_e = \frac{1}{3}$ pour tout e est une solution optimale) et toute solution de valeur égale à 5 vérifie $\sum_{e\in\delta(v)} x_e = 1$ pour tout $v \in V(G)$. Ainsi une solution demi-entière optimale doit être telle que $x_e = \frac{1}{2}$ pour les arêtes e d'un cycle hamiltonien et $x_e = 0$ sinon. Cependant, le graphe de Petersen n'est pas hamiltonien.

Néanmoins, la solution de la relaxation linéaire (20.12) conduit à un algorithme d'approximation de facteur 2. Il est essentiel de remarquer que pour toute solution de base optimale x, il existe une arête e telle que $x_e \geq \frac{1}{2}$ (théorème 20.33).

L'algorithme arrondira et fixera alors seulement ces composantes et considérera le problème restant qui a au moins une arête en moins.

Nous avons besoin de quelques résultats préliminaires. Pour un ensemble $S \subseteq V(G)$, on note χ_S le vecteur d'incidence de $\delta_G(S)$ respectivement à $E(G)$. Pour toute solution réalisable x de (20.12), on dit qu'un ensemble $S \subseteq V(G)$ est **serré** si $\chi_S x = f(S)$.

Lemme 20.32. (Jain [2001]) *Soient G un graphe, $m := |E(G)|$ et $f : 2^{V(G)} \to \mathbb{Z}_+$ une fonction faiblement supermodulaire. Considérons x une solution de base du PL (20.12) et supposons que $0 < x_e < 1$ pour chaque $e \in E(G)$. Alors il existe une famille laminaire $\mathcal{B}$ de m sous-ensembles serrés de $V(G)$ telle que les vecteurs χ_B, $B \in \mathcal{B}$, soient linéairement indépendants dans $\mathbb{R}^{E(G)}$.*

Preuve. Soit $\mathcal{B}$ une famille laminaire de sous-ensembles serrés de $V(G)$ telle que les vecteurs χ_B, $B \in \mathcal{B}$, soient linéairement indépendants. Supposons que $|\mathcal{B}| < m$; nous montrons comment étendre $\mathcal{B}$.

Puisque x est une solution de base de (20.12), c.-à-d. un sommet du polytope, il existe m contraintes d'inégalités linéairement indépendantes satisfaites avec égalité (proposition 3.9). Comme $0 < x_e < 1$ pour chaque $e \in E(G)$, ces contraintes correspondent à une famille $\mathcal{S}$ (non nécessairement laminaire) de m sous-ensembles serrés de $V(G)$ telle que les vecteurs χ_S ($S \in \mathcal{S}$) sont linéairement indépendants. Comme $|\mathcal{B}| < m$, il existe un ensemble serré $S \subseteq V(G)$ tel que les vecteurs χ_B, $B \in \mathcal{B} \cup \{S\}$ sont linéairement indépendants. Choisissons S tel que

$$\gamma(S) := |\{B \in \mathcal{B} : B \text{ croise } S\}|$$

soit minimal, où l'on dit que B croise S si $B \cap S \neq \emptyset$ et $B \setminus S \neq \emptyset$ et $S \setminus B \neq \emptyset$.

Si $\gamma(S) = 0$, alors nous pouvons ajouter S à $\mathcal{B}$ et nous avons terminé. Supposons donc que $\gamma(S) > 0$ et considérons un ensemble $B \in \mathcal{B}$ intersectant S. Comme f est faiblement supermodulaire, nous avons

$$\begin{aligned} f(S \setminus B) + f(B \setminus S) &\geq f(S) + f(B) \\ &= \sum_{e \in \delta_G(S)} x_e + \sum_{e \in \delta_G(B)} x_e \\ &= \sum_{e \in \delta_G(S \setminus B)} x_e + \sum_{e \in \delta_G(B \setminus S)} x_e + 2 \sum_{e \in E_G(S \cap B, V(G) \setminus (S \cup B))} x_e \end{aligned}$$

ou

$$\begin{aligned} f(S \cap B) + f(S \cup B) &\geq f(S) + f(B) \\ &= \sum_{e \in \delta_G(S)} x_e + \sum_{e \in \delta_G(B)} x_e \\ &= \sum_{e \in \delta_G(S \cap B)} x_e + \sum_{e \in \delta_G(S \cup B)} x_e + 2 \sum_{e \in E_G(S \setminus B, B \setminus S)} x_e. \end{aligned}$$

Dans le premier cas, $S \setminus B$ et $B \setminus S$ sont tous deux serrés et $E_G(S \cap B, V(G) \setminus (S \cup B)) = \emptyset$, ce qui implique $\chi_{S\setminus B} + \chi_{B\setminus S} = \chi_S + \chi_B$. Dans le second cas, $S \cap B$ et $S \cup B$ sont tous deux serrés et $E_G(S \setminus B, B \setminus S) = \emptyset$, ce qui implique $\chi_{S\cap B} + \chi_{S\cup B} = \chi_S + \chi_B$.

Ainsi, il y a au moins un ensemble T parmi $S \setminus B$, $B \setminus S$, $S \cap B$ et $S \cup B$ qui est serré et qui est tel que les vecteurs χ_B, $B \in \mathcal{B} \cup \{T\}$ sont linéairement indépendants. On montre finalement que $\gamma(T) < \gamma(S)$; cela fournit une contradiction au choix de S.

Puisque B croise S mais pas T, il suffit de montrer qu'il n'existe pas d'ensemble $C \in \mathcal{B}$ qui intersecte T mais pas S. En effet, comme T est l'un des ensembles $S \setminus B$, $B \setminus S$, $S \cap B$ et $S \cup B$, tout ensemble C qui intersecte T mais pas S doit intersecter B. Puisque $\mathcal{B}$ est laminaire et que $B \in \mathcal{B}$ cela implique $C \notin \mathcal{B}$. □

Nous pouvons maintenant prouver le théorème crucial pour l'algorithme de JAIN :

Théorème 20.33. (Jain [2001]) *Soient G un graphe et $f : 2^{V(G)} \to \mathbb{Z}_+$ une fonction faiblement supermodulaire qui n'est pas identiquement nulle. Soit x une solution de base du PL (20.12). Alors il existe une arête $e \in E(G)$ telle que $x_e \geq \frac{1}{2}$.*

Preuve. Nous pouvons supposer $x_e > 0$ pour chaque arête e, puisque sinon nous pouvons supprimer e. En fait, nous allons supposer $0 < x_e < \frac{1}{2}$ pour tout $e \in E(G)$ et en déduirons une contradiction.

D'après le lemme 20.32, il existe une famille laminaire $\mathcal{B}$ de $m := |E(G)|$ sous-ensembles serrés de $V(G)$ telle que les vecteurs χ_B, $B \in \mathcal{B}$, sont linéairement indépendants. L'indépendance linéaire implique en particulier qu'aucun des χ_B n'est le vecteur nul ; ainsi $0 < \chi_B x = f(B)$ et donc $f(B) \geq 1$ pour tout $B \in \mathcal{B}$. De plus, $\bigcup_{B\in\mathcal{B}} \delta_G(B) = E(G)$. D'après l'hypothèse $x_e < \frac{1}{2}$ pour tout $e \in E(G)$, nous avons $|\delta_G(B)| \geq 2f(B) + 1 \geq 3$ pour tout $B \in \mathcal{B}$.

Soit (T, φ) une représentation par arbre de $\mathcal{B}$. Pour chaque sommet t de l'arborescence T, nous notons T_t le sous-graphe maximal de T qui est une arborescence de racine t (T_t contient t et tous ses successeurs). De plus, définissons $B_t := \{v \in V(G) : \varphi(v) \in V(T_t)\}$. Par définition de la représentation par arbre, nous avons $B_r = V(G)$ pour la racine r de T et $\mathcal{B} = \{B_t : t \in V(T) \setminus \{r\}\}$.

Affirmation : pour tout $t \in V(T)$, nous avons $\sum_{v\in B_t} |\delta_G(v)| \geq 2|V(T_t)| + 1$ avec égalité seulement si $|\delta_G(B_t)| = 2f(B_t) + 1$.

Nous prouvons l'affirmation par induction sur $|V(T_t)|$. Si $\delta_T^+(t) = \emptyset$ (c.-à-d. $V(T_t) = \{t\}$), alors B_t est un élément minimal de $\mathcal{B}$ et donc $\sum_{v\in B_t} |\delta_G(v)| = |\delta_G(B_t)| \geq 3 = 2|V(T_t)| + 1$, avec égalité seulement si $|\delta_G(B_t)| = 3$ (ce qui implique $f(B_t) = 1$).

Pour l'induction, considérons $t \in V(T)$ tel que $\delta_T^+(t) \neq \emptyset$, disons $\delta_T^+(t) = \{(t, s_1), \ldots, (t, s_k)\}$, où k est le nombre de successeurs immédiats de t. Définissons $E_1 := \bigcup_{i=1}^k \delta_G(B_{s_i}) \setminus \delta_G(B_t)$ et $E_2 := \delta_G\left(B_t \setminus \bigcup_{i=1}^k B_{s_i}\right)$ (voir figure 20.5 pour une illustration).

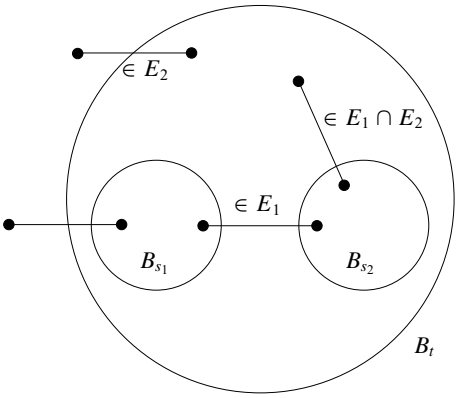


Figure 20.5.

Remarquons que $E_1 \cup E_2 \neq \emptyset$, car sinon $\chi_{B_t} = \sum_{i=1}^{k} \chi_{B_{s_i}}$, ce qui contredit l'hypothèse que les vecteurs χ_B, $B \in \mathcal{B}$ sont linéairement indépendants (remarquons que soit $B_t \in \mathcal{B}$, soit $t = r$ et alors $\chi_{B_t} = 0$). De plus, nous avons

$$|\delta_G(B_t)| + 2|E_1| \;=\; \sum_{i=1}^{k} |\delta_G(B_{s_i})| + |E_2| \tag{20.13}$$

et, puisque $B_{s_1}, \ldots, B_{s_k}$ et B_t sont serrés,

$$f(B_t) + 2 \sum_{e \in E_1} x_e \;=\; \sum_{i=1}^{k} f(B_{s_i}) + \sum_{e \in E_2} x_e. \tag{20.14}$$

De plus, par l'hypothèse d'induction,

$$\begin{aligned} \sum_{v \in B_t} |\delta_G(v)| \;&\geq\; \sum_{i=1}^{k} \sum_{v \in B_{s_i}} |\delta_G(v)| + |E_2| \\ &\geq\; \sum_{i=1}^{k} (2|V(T_{s_i})| + 1) + |E_2| \\ &=\; 2|V(T_t)| - 2 + k + |E_2|. \end{aligned} \tag{20.15}$$

Nous distinguons maintenant trois cas.

Cas 1 : $k + |E_2| \geq 3$. Alors d'après (20.15)

$$\sum_{v \in B_t} |\delta_G(v)| \;\geq\; 2|V(T_t)| + 1,$$

avec égalité seulement si $k + |E_2| = 3$ et $|\delta_G(B_{s_i})| = 2f(B_{s_i}) + 1$ pour $i = 1, \ldots, k$. Nous devons montrer qu'alors $|\delta_G(B_t)| = 2f(B_t) + 1$.

D'après (20.13), nous avons

$$|\delta_G(B_t)| + 2|E_1| \;=\; \sum_{i=1}^{k} |\delta_G(B_{s_i})| + |E_2| \;=\; 2\sum_{i=1}^{k} f(B_{s_i}) + k + |E_2|$$
$$\;=\; 2\sum_{i=1}^{k} f(B_{s_i}) + 3,$$

ainsi $|\delta_G(B_t)|$ est impair. Alors, à l'aide de (20.14), nous concluons que

$$|\delta_G(B_t)| + 2|E_1| \;=\; 2\sum_{i=1}^{k} f(B_{s_i}) + 3 \;=\; 2f(B_t) + 4\sum_{e\in E_1} x_e - 2\sum_{e\in E_2} x_e + 3$$
$$\;<\; 2f(B_t) + 2|E_1| + 3,$$

car $E_1 \cup E_2 \neq \emptyset$. Nous avons $|\delta_G(B_t)| = 2f(B_t) + 1$, comme requis.

Cas 2 : $k = 2$ et $E_2 = \emptyset$. Alors $E_1 \neq \emptyset$ et, d'après (20.14), $2\sum_{e\in E_1} x_e$ est un nombre entier, ainsi $2\sum_{e\in E_1} x_e \leq |E_1| - 1$. Remarquons que $E_1 \neq \delta_G(B_{s_1})$, car sinon $\chi_{B_{s_2}} = \chi_{B_{s_1}} + \chi_{B_t}$, ce qui contredit l'hypothèse que les vecteurs χ_B, $B \in \mathcal{B}$ sont linéairement indépendants. De manière similaire, $E_1 \neq \delta_G(B_{s_2})$. Pour $i = 1, 2$, nous obtenons

$$2f(B_{s_i}) \;=\; 2\sum_{e\in\delta(B_{s_i})\setminus E_1} x_e + 2\sum_{e\in E_1} x_e \;<\; |\delta_G(B_{s_i})\setminus E_1| + |E_1| - 1 \;=\; |\delta_G(B_{s_i})| - 1.$$

Par l'hypothèse d'induction, cela implique $\sum_{v\in B_{s_i}} |\delta_G(v)| \;>\; 2|V(T_{s_i})| + 1$ et comme dans (20.15) nous obtenons

$$\sum_{v\in B_t} |\delta_G(v)| \;\geq\; \sum_{i=1}^{2}\sum_{v\in B_{s_i}} |\delta_G(v)| \;\geq\; \sum_{i=1}^{2}(2|V(T_{s_i})| + 2)$$
$$\;=\; 2|V(T_t)| + 2.$$

Cas 3 : $k = 1$ et $|E_2| \leq 1$. Remarquons que $k = 1$ implique $E_1 \subseteq E_2$, ainsi $|E_2| = 1$. D'après (20.14), nous avons

$$\sum_{e\in E_2\setminus E_1} x_e - \sum_{e\in E_1} x_e \;=\; \sum_{e\in E_2} x_e - 2\sum_{e\in E_1} x_e \;=\; f(B_t) - f(B_{s_1}).$$

C'est une contradiction puisque le terme de droite est un nombre entier alors que le terme de gauche ne l'est pas. Donc le cas 3 ne peut pas se produire.

L'affirmation est prouvée. Pour $t = r$, nous obtenons $\sum_{v\in V(G)} |\delta_G(v)| \;\geq\; 2|V(T)| + 1$, c.-à-d. $2|E(G)| \;>\; 2|V(T)|$. Comme d'autre part $|V(T)| - 1 = |E(T)| = |\mathcal{B}| = |E(G)|$, nous avons une contradiction. □

D'après ce théorème, nous avons l'algorithme naturel suivant :

ALGORITHME DE JAIN

Input Un graphe non orienté G, des poids $c : E(G) \to \mathbb{R}_+$, des capacités $u : E(G) \to \mathbb{N}$ et une fonction propre $f : 2^{V(G)} \to \mathbb{Z}_+$ (donnée par un oracle).

Output Une fonction $x : E(G) \to \mathbb{Z}_+$ telle que $\sum_{e \in \delta_G(S)} x_e \geq f(S)$ pour tout $S \subseteq V(G)$.

① Pour tout $e \in E(G)$, poser $x_e := 0$ si $c(e) > 0$ et $x_e := u(e)$ si $c(e) = 0$.

② Trouver une solution de base optimale y du PL (20.12) respectivement à c, u' et f', où $u'(e) := u(e) - x_e$ pour tout $e \in E(G)$ et $f'(S) := f(S) - \sum_{e \in \delta_G(S)} x_e$ pour tout $S \subseteq V(G)$.
If $y_e = 0$ pour tout $e \in E(G)$, **then stop**.

③ Poser $x_e := x_e + \lceil y_e \rceil$ pour tout $e \in E(G)$ tel que $y_e \geq \frac{1}{2}$.
Go to ②.

Théorème 20.34. (Jain [2001]) *L'*ALGORITHME DE JAIN *trouve une solution entière du PL (20.12) dont le coût est au plus égal à deux fois la valeur optimale du PL. Il peut être implémenté de telle manière qu'il s'exécute en temps polynomial. Il s'agit ainsi d'un algorithme d'approximation de facteur 2 pour le* PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE*.*

Preuve. Après la première itération, on a $f'(S) \leq \sum_{e \in \delta_G(S)} \frac{1}{2} \leq \frac{|E(G)|}{2}$ pour tout $S \subseteq V(G)$. D'après le théorème 20.33 chaque itération suivante augmente au moins un x_e d'au moins 1 (remarquons que f est propre et donc faiblement supermodulaire d'après la proposition 20.18). Comme chaque x_e est augmenté d'au plus $\frac{|E(G)|}{2}$ après la première itération, le nombre total d'itérations est borné par $\frac{|E(G)|^2}{2}$.

La seule difficulté au niveau de l'implémentation est l'étape ②. D'après le théorème 4.21, il suffit de résoudre le PROBLÈME DE SÉPARATION. Pour un vecteur $y \in \mathbb{R}_+^{E(G)}$ donné, nous devons décider si $\sum_{e \in \delta_G(S)} y_e \geq f'(S) = f(S) - \sum_{e \in \delta_G(S)} x_e$ pour tout $S \subseteq V(G)$, et si ce n'est pas le cas, trouver une coupe violée. Comme f est propre, cela peut être réalisé en temps $O(n^4 + n\theta)$ d'après le théorème 20.20, où $n = |V(G)|$ et θ est la borne de l'oracle pour f.

Nous prouvons finalement que le rapport de performance est égal à 2, par induction sur le nombre d'itérations. Si l'algorithme se termine à la première itération, alors la solution est de coût nul et est donc optimale.

Sinon soient $x^{(1)}$ et $y^{(1)}$ les vecteurs x et y après la première itération et soit $x^{(t)}$ le vecteur x à la fin. Soit $z_e := y_e^{(1)}$ si $y_e^{(1)} < \frac{1}{2}$ et $z_e = 0$ sinon. On a $cx^{(1)} \leq 2c\left(y^{(1)} - z\right)$. Soit $f^{(1)}$ la fonction résiduelle définie par $f^{(1)}(S) := f(S) - \sum_{e \in \delta_G(S)} x_e^{(1)}$. Comme z est une solution réalisable pour $f^{(1)}$, nous savons d'après l'hypothèse d'induction que $c\left(x^{(t)} - x^{(1)}\right) \leq 2cz$. Nous concluons :

$$cx^{(t)} \;\leq\; cx^{(1)} + c\left(x^{(t)} - x^{(1)}\right) \;\leq\; 2c\left(y^{(1)} - z\right) + 2cz \;=\; 2cy^{(1)}.$$

Comme $cy^{(1)}$ est une borne inférieure du coût d'une solution optimale, nous avons terminé. □

Melkonian et Tardos [2004] ont étendu la technique de Jain à un problème de conception de réseau orienté. Fleischer, Jain et Williamson [2006] et Cheriyan et Vetta [2007] ont montré comment certaines contraintes de sommet-connexité peuvent être prises en compte. Cependant, des résultats dus à Kortsarz, Krauthgamer et Lee [2004] indiquent qu'il est difficile d'approximer la version sommet-connexe du PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE.

Exercices

1. Soit (G, c, T) une instance du PROBLÈME DE L'ARBRE DE STEINER où G est un graphe complet et $c : E(G) \to \mathbb{R}_+$ satisfait l'inégalité triangulaire. Prouver qu'il existe un arbre de Steiner optimum pour T avec au plus $|T| - 2$ points de Steiner.
2. Prouver que le PROBLÈME DE L'ARBRE DE STEINER est *MAXSNP*-difficile même pour les graphes complets avec des poids sur les arêtes égaux à 1 et 2 seulement.
   Indication : modifier la preuve du théorème 20.3. Que peut-on dire si G n'est pas connexe ?
   (Bern, Plassmann [1989])
3. Trouver un algorithme qui s'exécute en temps $O(n^3t^2)$ pour le PROBLÈME DE L'ARBRE DE STEINER dans les graphes planaires avec tous les terminaux situés sur la face extérieure et prouver son exactitude.
   Indication : montrer que dans l'ALGORITHME DE DREYFUS-WAGNER il suffit de considérer les ensembles $U \subseteq T$ qui sont consécutifs, c.-à-d. qu'il existe une chaîne P dont les sommets sont tous situés sur la face extérieure telle que $V(P) \cap T = U$ (nous supposons sans perte de généralité que G est 2-connexe).
   (Erickson, Monma et Veinott [1987])
4. Décrire un algorithme pour le PROBLÈME DE L'ARBRE DE STEINER qui s'exécute en temps $O(n^3)$ pour les instances (G, c, T) telles que $|V(G) \setminus T| \leq k$, où k est une constante.
5. Prouver le renforcement suivant du théorème 20.6 : si (G, c, T) est une instance du PROBLÈME DE L'ARBRE DE STEINER avec $|T| \geq 2$, $(\bar{G}, \bar{c})$ la fermeture métrique, S un arbre de Steiner optimum pour T dans G et M un arbre couvrant de poids minimum dans $\bar{G}[T]$ respectivement à $\bar{c}$, alors

   $$\bar{c}(M) \;\leq\; 2\left(1 - \frac{1}{b}\right) c(S),$$

   où b est le nombre de feuilles (sommets de degré 1) de S. Montrer que cette borne est serrée.

6. Prouver que le 4-Steiner ratio ρ_4 est égal à $\frac{3}{2}$.
7. Améliorer l'inégalité de la propositon 20.10 pour les cas $|T| = 3$ et $|T| = 4$.
8. Trouver un algorithme d'approximation de facteur 2 combinatoire pour le PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE avec $r_{ij} = k$ pour tout i, j (c.-à-d. le PROBLÈME DU SOUS-GRAPHE k-ARÊTE-CONNEXE DE POIDS MINIMUM).
   Indication : remplacer chaque arête par une paire d'arcs opposés (avec le même poids) et appliquer soit l'exercice 24 du chapitre 13 soit le théorème 6.17 directement.
   (Khuller et Vishkin [1994])
   Remarque : pour d'autres résultats concernant des problèmes similaires, voir Khuller et Raghavachari [1996], Gabow [2005], Jothi, Raghavachari et Varadarajan [2003] et Gabow *et al.* [2005].
9. Montrer que, dans le cas particulier du PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE, la relaxation fractionnaire (20.6) peut être formulée comme un PL de taille polynomiale.
10. Prouver le renforcement suivant du théorème 20.23. Étant donné une fonction propre g (donnée par un oracle) et un ensemble $F \subseteq E(G)$ satisfaisant presque g, les ensembles actifs respectivement à (g, F) peuvent être calculés avec une complexité $O\left(k^2n^2 + n^2\theta\right)$, où $n = |V(G)|$, $k = \max_{S \subseteq V(G)} g(S)$ et θ est le temps requis par l'oracle pour g.
    Indication : l'idée est d'arrêter les calculs de flots dès que la valeur du flot maximum est au moins égale à k, car les coupes ayant k arêtes ou plus ne sont pas intéressantes ici.
    L'ALGORITHME DE GOMORY-HU (voir paragraphe 8.6) est modifié de la façon suivante. À chaque étape, chaque sommet de l'arbre T est une forêt (au lieu d'un sous-ensemble de sommets). Les arêtes des forêts correspondent aux problèmes de flot maximum pour lesquels la valeur d'un flot maximum est au moins égale à k. À chaque itération de l'ALGORITHME DE GOMORY-HU modifié, nous sélectionnons deux sommets s et t dans des composantes connexes différentes de la forêt correspondant à un sommet de T. Si la valeur du flot maximum est au moins égale à k, on insère une arête (s, t) dans la forêt. Sinon, on divise le sommet comme dans la procédure classique de l'algorithme de Gomory-Hu. Nous arrêtons lorsque tous les sommets de T sont des arbres. Nous remplaçons finalement chaque sommet de T par l'arbre correspondant.
    Il est clair que l'arbre de Gomory-Hu modifié vérifie également les propriétés (20.8) et (20.9). Si les calculs de flots sont effectués à l'aide de l'ALGORITHME DE FORD-FULKERSON, que l'on arrête après la k-ième chaîne augmentante, alors on obtient la borne $O(k^2n^2)$.
    Remarque : l'ALGORITHME PRIMAL-DUAL POUR LA CONCEPTION DE RÉSEAU peut ainsi s'exécuter en temps $O\left(k^3n^3 + kn^3\theta\right)$.
    (Gabow, Goemans et Williamson [1998])

$*$ 11. Considérons le PROBLÈME DE CONCEPTION DE RÉSEAU FIABLE qui est, comme nous l'avons vu, un cas particulier de (20.1).

(a) Considérons un arbre couvrant maximum T dans le graphe complet avec des coûts r_{ij} sur les arêtes (i, j). Montrer que si un ensemble d'arêtes satisfait les conditions de connexité des arêtes de T, alors il satisfait toutes les conditions de connexité.

(b) Lorsque nous déterminons les ensembles actifs au début de la phase p, nous avons seulement besoin de rechercher une chaîne augmentante de i à j pour chaque $(i, j) \in E(T)$ (nous pouvons utiliser le flot de i à j de la phase précédente). S'il n'existe pas de chaîne augmentante de i à j, alors il y a au plus deux candidats pour les ensembles actifs. Parmi ces $O(n)$ candidats, nous pouvons trouver les ensembles actifs en temps $O(n^2)$.

(c) Montrer que la mise à jour de ces structures de données peut être effectuée en temps $O(kn^2)$ par phase.

(d) En conclure que les ensembles actifs peuvent être calculés en temps $O(k^2n^2)$.

(Gabow, Goemans et Williamson [1998])

12. Montrer que l'étape de mise à jour ⑤ de l'ALGORITHME PRIMAL-DUAL POUR LA CONCEPTION DE RÉSEAU est cruciale : sans l'étape ⑤, l'algorithme ne réalise même pas un rapport de performance fini pour $k = 1$.

13. On ne connaît pas d'algorithme pour le PROBLÈME DU T-JOINT DE POIDS MINIMUM ayant une complexité inférieure à $O(n^3)$ pour les graphes denses (voir corollaire 12.11). Soit G un graphe non orienté, $c : E(G) \to \mathbb{R}_+$ et $T \subseteq V(G)$ avec $|T|$ pair. Considérons le PL en nombres entiers (20.1), où l'on pose $f(S) := 1$ si $|S \cap T|$ est impair et $f(S) := 0$ sinon.

(a) Prouver que notre algorithme primal-dual appliqué à (20.1) retourne une forêt dans laquelle chaque composante connexe contient un nombre pair d'éléments de T.

(b) Prouver qu'une solution optimale de (20.1) est un T-joint de poids minimum plus éventuellement quelques arêtes de poids nul.

(c) L'algorithme primal-dual peut être implémenté en temps $O(n^2 \log n)$ si $f(S) \in \{0, 1\}$ pour tout S. Montrer que cela implique un algorithme d'approximation de facteur 2 pour le PROBLÈME DU T-JOINT DE POIDS MINIMUM avec des poids non négatifs, ayant le même temps de calcul.
Indication : d'après (a), l'algorithme retourne une forêt F. Pour chaque composante connexe C de F, considérer $\bar{G}[V(C) \cap T]$ et trouver un tour dont le poids est au plus égal à deux fois le poids de C (voir la preuve du théorème 20.6). Ensuite prendre une arête sur deux dans le tour. (Une idée similaire est à la base de l'ALGORITHME DE CHRISTOFIDES ; voir paragraphe 21.1.)

(Goemans et Williamson [1995])

14. Trouver une solution de base optimale x de (20.12), où G est le graphe de Petersen (figure 20.4) et $f(S) = 1$ pour tout $0 \neq S \subset V(G)$. Trouver une famille laminaire maximale $\mathcal{B}$ d'ensembles serrés respectivement à x telle que les vecteurs χ_B, $B \in \mathcal{B}$, soient linéairement indépendants (voir lemme 20.32).

15. Prouver que la valeur optimale de (20.12) peut être arbitrairement proche de la moitié de la valeur d'une solution entière optimale.
Remarque : d'après l'ALGORITHME DE JAIN (voir la preuve du théorème 20.34), cela ne peut pas être inférieur à la moitié.

Références

Littérature générale :

Cheng, X., Du, D.-Z. [2001] : Steiner Trees in Industry. Kluwer, Dordrecht 2001

Du, D.-Z., Smith, J.M., Rubinstein, J.H. [2000] : Advances in Steiner Trees. Kluwer, Boston 2000

Hwang, F.K., Richards, D.S., Winter, P. [1992] : The Steiner Tree Problem ; Annals of Discrete Mathematics 53. North-Holland, Amsterdam 1992

Goemans, M.X., Williamson, D.P. [1996] : The primal-dual method for approximation algorithms and its application to network design problems. In : Approximation Algorithms for *NP*-Hard Problems. (D.S. Hochbaum, ed.), PWS, Boston, 1996

Grötschel, M., Monma, C.L., Stoer, M. [1995] : Design of survivable networks. In : Handbooks in Operations Research and Management Science ; Volume 7 ; Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995

Kerivin, H., Mahjoub, A.R. [2005] : Design of survivable networks : a survey. Networks 46 (2005), 1–21

Prömel, H.J., Steger, A. [2002] : The Steiner Tree Problem. Vieweg, Braunschweig 2002

Stoer, M. [1992] : Design of Survivable Networks. Springer, Berlin 1992

Vazirani, V.V. [2001] : Approximation Algorithms. Springer, Berlin 2001, Chapters 22 and 23

Références citées :

Agrawal, A., Klein, P.N., Ravi, R. [1995] : When trees collide : an approximation algorithm for the generalized Steiner tree problem in networks. SIAM Journal on Computing 24 (1995), 440–456

Arora, S. [1998] : Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. Journal of the ACM 45 (1998), 753–782

Berman, P., Ramaiyer, V. [1994] : Improved approximations for the Steiner tree problem. Journal of Algorithms 17 (1994), 381–408

Bern, M., Plassmann, P. [1989] : The Steiner problem with edge lengths 1 and 2. Information Processing Letters 32 (1989), 171–176

Bertsimas, D., Teo, C. [1995] : From valid inequalities to heuristics : a unified view of primal-dual approximation algorithms in covering problems. Operations Research 46 (1998), 503–514

Bertsimas, D., Teo, C. [1997] : The parsimonious property of cut covering problems and its applications. Operations Research Letters 21 (1997), 123–132

Borchers, A., Du, D.-Z. [1997] : The k-Steiner ratio in graphs. SIAM Journal on Computing 26 (1997), 857–869

Borradaile, G., Kenyon-Mathieu, C., Klein, P. [2007] : A polynomial-time approximation scheme for Steiner tree in planar graphs. Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (2007), 1285–1294

Cheriyan, J., Vetta, A. [2007] : Approximation algorithms for network design with metric costs. SIAM Journal on Discrete Mathematics 21 (2007), 612–636

Choukhmane, E. [1978] : Une heuristique pour le problème de l'arbre de Steiner. RAIRO Recherche Opérationnelle 12 (1978), 207–212 [in French]

Clementi, A.E.F., Trevisan, L. [1999] : Improved non-approximability results for minimum vertex cover with density constraints. Theoretical Computer Science 225 (1999), 113–128

Dreyfus, S.E., Wagner, R.A. [1972] : The Steiner problem in graphs. Networks 1 (1972), 195–207

Du, D.-Z., Zhang, Y., Feng, Q. [1991] : On better heuristic for Euclidean Steiner minimum trees. Proceedings of the 32nd Annual Symposium on the Foundations of Computer Science (1991), 431–439

Erickson, R.E., Monma, C.L., Veinott, A.F., Jr. [1987] : Send-and-split method for minimum concave-cost network flows. Mathematics of Operations Research 12 (1987), 634–664

Fleischer, L., Jain, K., Williamson, D.P. [2006] : Iterative rounding 2-approximation algorithms minimum-cost vertex connectivity problems. Journal of Computer and System Sciences 72 (2006), 838–867

Fuchs, B., Kern, W., Mölle, D., Richter, S., Rossmanith, P., Wang, X. [2007] : Dynamic programming for minimum Steiner trees. Theory of Computing Systems 41 (2007), 493–500

Gabow, H.N. [2005] : An improved analysis for approximating the smallest k-edge connected spanning subgraph of a multigraph. SIAM Journal on Discrete Mathematics 19 (2005), 1–18

Gabow, H.N., Goemans, M.X., Williamson, D.P. [1998] : An efficient approximation algorithm for the survivable network design problem. Mathematical Programming B 82 (1998), 13–40

Gabow, H.N., Goemans, M.X., Tardos, É., Williamson, D.P. [2005] : Approximating the smallest k-edge connected spanning subgraph by LP-rounding. Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (2005), 562–571

Garey, M.R., Graham, R.L., Johnson, D.S. [1977] : The complexity of computing Steiner minimal trees. SIAM Journal of Applied Mathematics 32 (1977), 835–859

Garey, M.R., Johnson, D.S. [1977] : The rectilinear Steiner tree problem is *NP*-complete. SIAM Journal on Applied Mathematics 32 (1977), 826–834

Gilbert, E.N., Pollak, H.O. [1968] : Steiner minimal trees. SIAM Journal on Applied Mathematics 16 (1968), 1–29

Goemans, M.X., Bertsimas, D.J. [1993] : Survivable networks, linear programming and the parsimonious property, Mathematical Programming 60 (1993), 145–166

Goemans, M.X., Goldberg, A.V., Plotkin, S., Shmoys, D.B., Tardos, É., Williamson, D.P. [1994] : Improved approximation algorithms for network design problems. Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (1994), 223–232

Goemans, M.X., Williamson, D.P. [1995] : A general approximation technique for constrained forest problems. SIAM Journal on Computing 24 (1995), 296–317

Gröpl, C., Hougardy, S., Nierhoff, T., Prömel, H.J. [2001] : Approximation algorithms for the Steiner tree problem in graphs. In : Cheng Du [2001], pp. 235–279

Hanan, M. [1966] : On Steiner's problem with rectilinear distance. SIAM Journal on Applied Mathematics 14 (1966), 255–265

Hetzel, A. [1995] : Verdrahtung im VLSI-Design : Spezielle Teilprobleme und ein sequentielles Lösungsverfahren. Ph.D. thesis, University of Bonn, 1995 [in German]

Hougardy, S., Prömel, H.J. [1999] : A 1.598 approximation algorithm for the Steiner tree problem in graphs. Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (1999), 448–453

Hwang, F.K. [1976] : On Steiner minimal trees with rectilinear distance. SIAM Journal on Applied Mathematics 30 (1976), 104–114

Jain, K. [2001] : A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica 21 (2001), 39–60

Jothi, R., Raghavachari, B., Varadarajan, S. [2003] : A 5/4-approximation algorithm for minimum 2-edge-connectivity. Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (2003), 725–734

Karp, R.M. [1972] : Reducibility among combinatorial problems. In : Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, eds.), Plenum Press, New York 1972, pp. 85–103

Karpinski, M., Zelikovsky, A. [1997] : New approximation algorithms for Steiner tree problems. Journal of Combinatorial Optimization 1 (1997), 47–65

Khuller, S., Raghavachari, B. [1996] : Improved approximation algorithms for uniform connectivity problems. Journal of Algorithms 21 (1996), 434–450

Khuller, S., Vishkin, U. [1994] : Biconnectivity augmentations and graph carvings. Journal of the ACM 41 (1994), 214–235

Klein, P.N., Ravi, R. [1993] : When cycles collapse : a general approximation technique for constrained two-connectivity problems. Proceedings of the 3rd Integer Programming and Combinatorial Optimization Conference (1993), 39–55

Korte, B., Prömel, H.J., Steger, A. [1990] : Steiner trees in VLSI-layout. In : Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, eds.), Springer, Berlin 1990, pp. 185–214

Kortsarz, G., Krauthgamer, R., Lee, J.R. [2004] : Hardness of approximation for vertex-connectivity network design problems. SIAM Journal on Computing 33 (2004), 704–720

Kou, L. [1990] : A faster approximation algorithm for the Steiner problem in graphs. Acta Informatica 27 (1990), 369–380

Kou, L., Markowsky, G., Berman, L. [1981] : A fast algorithm for Steiner trees. Acta Informatica 15 (1981), 141–145

Martin, A. [1992] : Packen von Steinerbäumen : Polyedrische Studien und Anwendung. Ph.D. thesis, Technical University of Berlin 1992 [in German]

Mehlhorn, K. [1988] : A faster approximation algorithm for the Steiner problem in graphs. Information Processing Letters 27 (1988), 125–128

Melkonian, V., Tardos, É. [2004] : Algorithms for a network design problem with crossing supermodular demands. Networks 43 (2004), 256–265

Robins, G., Zelikovsky, A. [2005] : Tighter bounds for graph Steiner tree approximation. SIAM Journal on Discrete Mathematics 19 (2005), 122–134

Takahashi, M., Matsuyama, A. [1980] : An approximate solution for the Steiner problem in graphs. Mathematica Japonica 24 (1980), 573–577

Thimm, M. [2003] : On the approximability of the Steiner tree problem. Theoretical Computer Science 295 (2003), 387–402

Warme, D.M., Winter, P., Zachariasen, M. [2000] : Exact algorithms for plane Steiner tree problems : a computational study. In : Advances in Steiner trees (D.-Z. Du, J.M. Smith, J.H. Rubinstein, eds.), Kluwer Academic Publishers, Boston, 2000, pp. 81–116

Williamson, D.P., Goemans, M.X., Mihail, M., Vazirani, V.V. [1995] : A primal-dual approximation algorithm for generalized Steiner network problems. Combinatorica 15 (1995), 435–454

Zelikovsky, A.Z. [1993] : An 11/6-approximation algorithm for the network Steiner problem. Algorithmica 9 (1993), 463–470

Chapitre 21

Le problème du voyageur de commerce

Au chapitre 15, nous avons déjà présenté le PROBLÈME DU VOYAGEUR DE COMMERCE (PVC) et montré qu'il est *NP*-difficile (théorème 15.42). Le PVC est sans doute le problème d'optimisation combinatoire *NP*-difficile le plus étudié et de nombreuses techniques ont été proposées. Nous commençons par la présentation des algorithmes d'approximation aux paragraphes 21.1 et 21.2. En pratique, les algorithmes dits de recherche locale (présentés au paragraphe 21.3) trouvent de meilleures solutions pour les instances de grande taille bien qu'ils n'aient pas un rapport fini de performance.

Nous étudions le polytope du voyageur de commerce (l'enveloppe convexe des vecteurs d'incidence de tous les tours dans K_n) au paragraphe 21.4. En utilisant une approche par plans coupants (voir paragraphe 5.5) combinée à une méthode du type séparation et évaluation (en anglais *branch and bound*), on peut résoudre optimalement des instances du PVC avec plusieurs milliers de villes. Nous présenterons cela au paragraphe 21.6, après avoir montré comment obtenir de bonnes bornes inférieures au paragraphe 21.5. Remarquons que toutes ces idées et techniques peuvent également être appliquées à d'autres problèmes d'optimisation combinatoire. Nous les présentons dans le cadre du PVC, car il s'agit d'un problème pour lequel ces techniques se sont révélées être très efficaces.

Nous considérons seulement le PVC symétrique, bien que le problème du voyageur de commerce asymétrique (dans lequel la distance de i à j peut être différente de la distance de j à i) soit aussi intéressant (voir exercice 4).

21.1 Algorithmes d'approximation pour le PVC

Dans ce paragraphe et le suivant, nous nous intéressons à l'approximabilité du PVC. Nous commençons par le résultat négatif suivant :

Théorème 21.1. (Sahni et Gonzalez [1976]) *À moins que P = NP, il n'existe pas d'algorithme d'approximation de facteur k pour le* PVC *quel que soit $k \geq 1$.*

Preuve. Supposons qu'il existe un algorithme A d'approximation de facteur k pour le PVC. Nous prouvons qu'il existe alors un algorithme polynomial pour le PROBLÈME DU CYCLE HAMILTONIEN. Comme ce dernier est *NP*-complet (d'après le théorème 15.25), cela implique *P = NP*.

Étant donné un graphe G, nous construisons une instance du PVC avec $n = |V(G)|$ villes : les distances sont définies par $c : E(K_n) \to \mathbb{Z}_+$,

$$c((i,j)) := \begin{cases} 1 & \text{si } (i,j) \in E(G) \\ 2 + (k-1)n & \text{si } (i,j) \notin E(G). \end{cases}$$

Nous appliquons maintenant l'algorithme A à cette instance. Si le tour retourné par l'algorithme est de longueur n, alors ce tour est un cycle hamiltonien de G. Sinon le tour retourné est de longueur au moins égale à $n + 1 + (k-1)n = kn + 1$, où $n := |V(G)|$. Si $\mathrm{OPT}(K_n, c)$ est la longueur du tour optimum, alors $\frac{kn+1}{\mathrm{OPT}(K_n,c)} \leq k$ puisque A est un algorithme d'approximation de facteur k. Ainsi $\mathrm{OPT}(K_n, c) > n$, ce qui montre que G n'a pas de cycle hamiltonien. □

Dans la plupart des applications pratiques, les distances considérées dans les instances du PVC vérifient l'inégalité triangulaire :

PVC MÉTRIQUE	
Instance	Un graphe complet K_n avec des poids $c : E(K_n) \to \mathbb{R}_+$ tels que $c((x,y)) + c((y,z)) \geq c((x,z))$ pour tout $x, y, z \in V(K_n)$.
Tâche	Trouver un cycle hamiltonien dans K_n de poids minimum.

En d'autres termes, (K_n, c) est sa propre fermeture métrique.

Théorème 21.2. *Le* PVC MÉTRIQUE *est NP-difficile.*

Preuve. Transformation à partir du PROBLÈME DU CYCLE HAMILTONIEN comme dans la preuve du théorème 15.42. □

On peut immédiatement penser à plusieurs heuristiques pour générer des solutions relativement bonnes. L'une des plus simples d'entre elles est l'heuristique du plus proche voisin : étant donné une instance (K_n, c) du PVC, choisissons $v_1 \in V(K_n)$ arbitrairement. Ensuite, pour $i = 2, \ldots, n$, choisissons v_i parmi $V(K_n) \setminus \{v_1, \ldots, v_{i-1}\}$ tel que $c((v_{i-1}, v_i))$ soit minimum. Autrement dit, on choisit à chaque étape la ville non visitée la plus proche.

L'heuristique du plus proche voisin n'est pas un algorithme d'approximation pour le PVC MÉTRIQUE. Pour un nombre infini de n, il existe des instances (K_n, c) pour lesquelles l'heuristique du plus proche voisin retourne un tour de longueur $\frac{1}{3}\,\mathrm{OPT}(K_n, c) \log n$ (Rosenkrantz, Stearns et Lewis [1977]). Voir aussi Hurkens et Woeginger [2004].

Le reste de ce paragraphe est consacré aux algorithmes d'approximation pour le PVC MÉTRIQUE. Ces algorithmes construisent d'abord une chaîne fermée contenant tous les sommets (mais avec répétition éventuelle de certains sommets). Comme le lemme suivant le montre, cela est suffisant si l'inégalité triangulaire est vérifiée.

Lemme 21.3. *Étant donné une instance (K_n, c) du* PVC MÉTRIQUE *et un graphe eulérien connexe G couvrant $V(K_n)$, avec éventuellement des arêtes parallèles. On peut alors construire un tour de poids au plus égal à $c(E(G))$ en temps linéaire.*

Preuve. D'après le théorème 2.25, on peut trouver un parcours eulérien dans G en temps linéaire. L'ordre dans lequel les sommets apparaissent dans ce cycle (on ne tient compte que de la première occurence d'un sommet) définit un tour. L'inégalité triangulaire implique immédiatement que la longueur de ce tour est inférieure ou égale à $c(E(G))$. □

Nous avons déjà rencontré cette idée dans le cas de l'approximation du PROBLÈME DE L'ARBRE DE STEINER (théorème 20.6).

ALGORITHME ARBRE-DOUBLE

Input Une instance (K_n, c) du PVC MÉTRIQUE.

Output Un tour.

① Trouver un arbre couvrant de poids minimum T dans K_n respectivement à c.

② Soit G le graphe contenant deux copies de chaque arête de T. G vérifie les prérequis du lemme 21.3.
Construire un tour comme dans la preuve du lemme 21.3.

Théorème 21.4. *L'*ALGORITHME ARBRE-DOUBLE *est un algorithme d'approximation de facteur 2 pour le* PVC MÉTRIQUE. *Son temps de calcul est $O(n^2)$.*

Preuve. Le temps de calcul est une conséquence du théorème 6.5. Nous avons $c(E(T)) \leq \mathrm{OPT}(K_n, c)$, puisqu'en supprimant une arête d'un tour, nous obtenons un arbre couvrant. Ainsi $c(E(G)) = 2c(E(T)) \leq 2\,\mathrm{OPT}(K_n, c)$. Le théorème est alors une conséquence du lemme 21.3. □

Pour les instances euclidiennes, on peut trouver un tour optimum dans le graphe G à l'étape ② en temps $O(n^3)$ au lieu d'appliquer le lemme 21.3 (Burkard, Deĭneko et Woeginger [1998]). La garantie de performance de l'ALGORITHME ARBRE-DOUBLE est serrée (exercice 5). Le meilleur algorithme d'approximation connu pour le PVC MÉTRIQUE est dû à Christofides [1976] :

ALGORITHME DE CHRISTOFIDES

Input Une instance (K_n, c) du PVC MÉTRIQUE.

Output Un tour.

① Trouver un arbre couvrant de poids minimum T dans K_n respectivement à c.
② Soit W l'ensemble des sommets de degré impair dans T.
Trouver un W-joint de poids minimum J dans K_n respectivement à c.
③ Soit $G := (V(K_n), E(T) \cup J)$. G vérifie les prérequis du lemme 21.3.
Construire un tour comme dans la preuve du lemme 21.3.

Une conséquence de l'inégalité triangulaire est que l'on peut prendre un couplage parfait de poids minimum dans $K_n[W]$ à la place de J à l'étape ②.

Théorème 21.5. (Christofides [1976]) *L'*ALGORITHME DE CHRISTOFIDES *est un algorithme d'approximation de facteur $\frac{3}{2}$ pour le* PVC MÉTRIQUE. *Son temps de calcul est $O(n^3)$.*

Preuve. La borne est une conséquence du théorème 12.9. Comme dans la preuve du théorème 21.4, nous avons $c(E(T)) \leq \mathrm{OPT}(K_n, c)$. Puisque chaque tour est l'union de deux W-joints, nous avons également $c(J) \leq \frac{1}{2}\,\mathrm{OPT}(K_n, c)$. Nous en concluons que $c(E(G)) = c(E(T)) + c(J) \leq \frac{3}{2}\,\mathrm{OPT}(K_n, c)$ et le résultat est une conséquence du lemme 21.3. □

On ne sait pas s'il existe un algorithme d'approximation ayant un meilleur rapport de performance. D'autre part, nous avons le résultat négatif suivant :

Théorème 21.6. (Papadimitriou et Yannakakis [1993]) *Le* PVC MÉTRIQUE *est MAXSNP-difficile.*

Preuve. Nous décrivons une L-réduction du PROBLÈME DE LA COUVERTURE MINIMUM PAR LES SOMMETS pour les graphes de degré maximum égal à 4 (qui est *MAXSNP*-difficile d'après le théorème 16.46) au PVC MÉTRIQUE.

Étant donné un graphe non orienté G de degré maximum égal à 4, nous construisons une instance (H, c) du PVC MÉTRIQUE de la façon suivante :

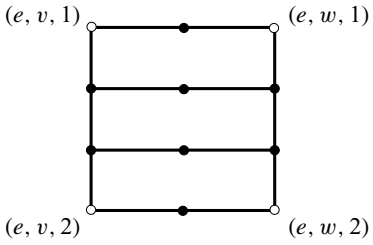


Figure 21.1.

Pour chaque $e = (v, w) \in E(G)$, nous introduisons un sous-graphe H_e, représenté à la figure 21.1, ayant douze sommets et quatorze arêtes. Quatre sommets de H_e, notés $(e, v, 1)$, $(e, v, 2)$, $(e, w, 1)$ et $(e, w, 2)$, vérifient une propriété

particulière. Le graphe H_e est tel qu'il y a une chaîne hamiltonienne de $(e, v, 1)$ à $(e, v, 2)$ et une autre de $(e, w, 1)$ à $(e, w, 2)$, mais il n'y a pas de chaîne hamiltonienne de (e, v, i) à (e, w, j) pour tout $i, j \in \{1, 2\}$.

H est maintenant le graphe complet sur l'ensemble de sommets $V(H) := \bigcup_{e \in E(G)} V(H_e)$. Pour $(x, y) \in E(H)$, nous posons

$$c((x,y)) := \begin{cases} 1 & \text{si } (x,y) \in E(H_e) \text{ pour une arête } e \in E(G)\,; \\ \mathrm{dist}_{H_e}(x,y) & \text{si } x, y \in V(H_e) \text{ pour une arête } e \in E(G) \\ & \text{mais } (x,y) \notin E(H_e)\,; \\ 4 & \text{si } x = (e,v,i) \text{ et } y = (f,v,j) \text{ avec } e \neq f\,; \\ 5 & \text{sinon.} \end{cases}$$

Cette construction est illustrée par la figure 21.2 (seules les arêtes de longueur égale à 1 ou 4 sont représentées).

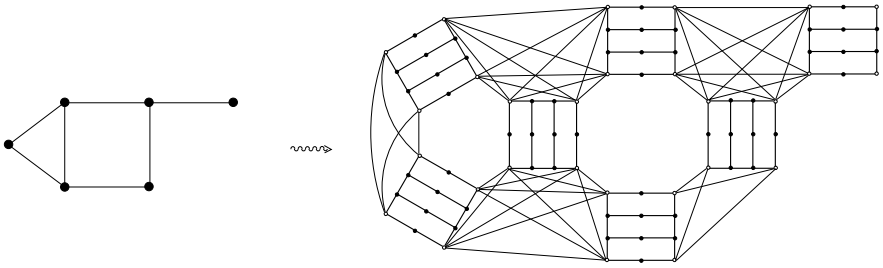


Figure 21.2.

(H, c) est une instance du PVC MÉTRIQUE. Nous affirmons qu'elle vérifie les propriétés suivantes :

(a) Pour chaque couverture par les sommets X de G, il y a un tour de longueur $15|E(G)| + |X|$.

(b) Étant donné un tour T, on peut construire un autre tour T' en temps polynomial qui est au plus aussi long et contient une chaîne hamiltonienne de chaque sous-graphe H_e ($e \in E(G)$).

(c) Étant donné un tour de longueur $15|E(G)| + k$, on peut construire une couverture par les sommets de cardinalité k dans G en temps polynomial.

(a) et (c) impliquent que nous avons une L-réduction, car la longueur du tour optimum est $15|E(G)| + \tau(G) \leq 15(4\tau(G)) + \tau(G)$ puisque G est de degré maximum égal à 4.

Afin de prouver (a), considérons une couverture par les sommets X de G et une partition $(E_x)_{x \in X}$ de $E(G)$ telle que $E_x \subseteq \delta(x)$ ($x \in X$). Alors pour chaque $x \in X$, le sous-graphe induit par $\bigcup_{e \in E_x} V(H_e)$ contient évidemment une chaîne hamiltonienne avec $11|E_x|$ arêtes de longueur 1 et $|E_x| - 1$ arêtes de longueur 4. En

ajoutant $|X|$ arêtes à l'union de ces chaînes hamiltoniennes, on obtient un tour ayant seulement $|X|$ arêtes de longueur 5, $|E(G)| - |X|$ arêtes de longueur 4 et $11|E(G)|$ arêtes de longueur 1.

Afin de prouver (b), considérons un tour T et $e \in E(G)$ tel que T ne contienne pas une chaîne hamiltonienne de H_e. Soit $(x, y) \in E(T)$, $x \notin V(H_e)$, $y \in V(H_e)$, et soit z le premier sommet atteint en dehors de $V(H_e)$ lorsque l'on parcourt T à partir de y sans passer par x. On supprime alors la partie du tour comprise entre x et z et on la remplace par l'arête $(x, (e, v, 1))$, une chaîne hamiltonienne de H_e reliant $(e, v, 1)$ à $(e, v, 2)$ et l'arête $((e, v, 2), z)$ (où $v \in e$ est choisi arbitrairement). À tous les autres endroits où T contient des sommets de H_e, on raccourcit T. Nous affirmons que le tour obtenu T' n'est pas plus long que T.

Supposons d'abord que $k := |\delta_T(V(H_e))| \geq 4$. Alors le poids total des arêtes de T incidentes à $V(H_e)$ est au moins égal à $4k + (12 - \frac{k}{2})$. Dans T', le poids total des arêtes incidentes à $V(H_e)$ est au plus égal à $5 + 5 + 11$ et nous avons également ajouté $\frac{k}{2} - 1$ arêtes lorsque nous avons raccourci T. Comme $5+5+11+5(\frac{k}{2}-1) \leq 4k + (12 - \frac{k}{2})$, le tour n'est pas devenu plus long.

Supposons maintenant que $|\delta_T(V(H_e))| = 2$, mais que T contient une arête (x, y) telle que $x, y \in V(H_e)$ mais $(x, y) \notin E(H_e)$. Alors la longueur totale des arêtes de T incidentes à $V(H_e)$ est au moins égale à 21, comme on peut facilement le vérifier. Comme la longueur totale des arêtes de T' incidentes à $V(H_e)$ est au plus égale à $5 + 5 + 11 = 21$, le tour n'est pas devenu plus long.

Nous prouvons finalement (c). Soit T un tour de longueur $15|E(G)| + k$, pour un certain k. D'après (b), nous pouvons supposer que T contient une chaîne hamiltonienne de chaque H_e ($e \in E(G)$), disons de $(e, v, 1)$ à $(e, v, 2)$; nous posons ici $v(e) := v$. Alors $X := \{v(e) : e \in E(G)\}$ est une couverture par les sommets de G. Comme T contient exactement $11|E(G)|$ arêtes de longueur 1, $|E(G)|$ arêtes de longueur 4 ou 5 et au moins $|X|$ arêtes de longueur 5, nous en concluons que $|X| \leq k$. □

Donc, d'après le corollaire 16.40, il ne peut exister de schéma d'approximation à moins que *P = NP*. Papadimitriou et Vempala [2006] ont montré que même l'existence d'un algorithme d'approximation de facteur $\frac{220}{219}$ impliquerait *P = NP*.

Papadimitriou et Yannakakis [1993] ont prouvé que le problème reste *MAXSNP*-difficile même si tous les poids sont égaux à 1 ou 2. Pour ce cas particulier, Berman et Karpinski [2006] ont trouvé un algorithme d'approximation de facteur $\frac{8}{7}$.

21.2 Problème du voyageur de commerce euclidien

Dans ce paragraphe, nous considérons le PVC pour des instances euclidiennes.

PVC EUCLIDIEN	
Instance	Un ensemble fini $V \subseteq \mathbb{R}^2$, $\lvert V\rvert \geq 3$.
Tâche	Trouver un cycle hamiltonien T dans le graphe complet sur V tel que la longueur totale $\sum_{(v,w)\in E(T)} \lvert\lvert v-w\rvert\rvert_2$ soit minimum.

On note ici $||v-w||_2$ la distance euclidienne entre v et w. On identifiera souvent une arête au segment de droite qui joint ses deux extrémités. Tout cycle hamiltonien optimal peut ainsi être considéré comme un polygone (il ne peut pas y avoir de croisement).

Le PVC EUCLIDIEN est évidemment un cas particulier du PVC MÉTRIQUE, et il est également fortement *NP*-difficile (Garey, Graham et Johnson [1976], Papadimitriou [1977]). Cependant, on peut utiliser la nature géométrique du problème de la manière suivante.

Supposons que nous ayons un ensemble de n points dans le carré unitaire, partitionné par une grille régulière telle que chaque région contienne peu de points. Nous cherchons alors un tour optimum dans chaque région et raccordons ensuite ces tours ensemble. Cette méthode a été proposée par Karp [1977] qui a montré qu'elle fournit des solutions approchées à un facteur $(1+\epsilon)$ près pour presque toutes les instances générées aléatoirement dans le carré unitaire. Arora [1998] a approfondi cette idée et a trouvé un schéma d'approximation pour le PVC EUCLIDIEN, que nous présentons dans ce paragraphe. Un schéma d'approximation similaire a été proposé par Mitchell [1999].

Dans ce paragraphe, nous fixons $0 < \epsilon < 1$. Nous montrons comment trouver en temps polynomial un tour dont la longueur est au plus égale à la longueur d'un tour optimal augmentée d'un facteur $1+\epsilon$. Nous commençons par arrondir les coordonnées :

Définition 21.7. *Une instance $V \subseteq \mathbb{R}^2$ du* PVC EUCLIDIEN *est dite* **correctement arrondie** *si les conditions suivantes sont vérifiées :*

(a) *Pour tout $(v_x, v_y) \in V$, v_x et v_y sont des nombres impairs.*
(b) $\max_{v,w\in V} ||v-w||_2 \leq \frac{64n}{\epsilon} + 16$, *où* $n := |V|$.
(c) $\min_{v,w\in V} ||v-w||_2 \geq 8$.

Le résultat suivant dit qu'il est suffisant de traiter des instances correctement arrondies :

Proposition 21.8. *Supposons qu'il existe un schéma d'approximation pour le* PVC EUCLIDIEN *restreint aux instances correctement arrondies. Alors il existe un schéma d'approximation pour le* PVC EUCLIDIEN *général.*

Preuve. Soient $V \subseteq \mathbb{R}^2$ un ensemble fini et $n := |V| \geq 3$. Définissons $L := \max_{v,w\in V} ||v-w||_2$ et

$$V' := \left\{ \left(1 + 8 \left\lfloor \frac{8n}{\epsilon L} v_x \right\rfloor, 1 + 8 \left\lfloor \frac{8n}{\epsilon L} v_y \right\rfloor \right) : (v_x, v_y) \in V \right\}.$$

V' peut contenir moins d'éléments que V. Comme la distance maximale dans V' est au plus égale à $\frac{64n}{\epsilon} + 16$, V' est correctement arrondi. Nous exécutons le schéma d'approximation (nous supposons qu'il existe) pour V' et $\frac{\epsilon}{2}$ et nous obtenons un tour dont la longueur l' est au plus égale à $(1 + \frac{\epsilon}{2}) \operatorname{OPT}(V')$.

À partir de ce tour, nous construisons un tour pour l'instance de départ V de manière directe. La longueur l de ce tour n'est pas supérieure à $\left(\frac{l'}{8} + 2n\right) \frac{\epsilon L}{8n}$. De plus,

$$\operatorname{OPT}(V') \;\leq\; 8\left(\frac{8n}{\epsilon L} \operatorname{OPT}(V) + 2n\right).$$

Nous avons donc

$$l \leq \frac{\epsilon L}{8n}\left(2n + \left(1 + \frac{\epsilon}{2}\right)\left(\frac{8n}{\epsilon L} \operatorname{OPT}(V) + 2n\right)\right) = \left(1 + \frac{\epsilon}{2}\right) \operatorname{OPT}(V) + \frac{\epsilon L}{2} + \frac{\epsilon^2 L}{8}.$$

Comme $\operatorname{OPT}(V) \geq 2L$, nous en concluons que $l \leq (1 + \epsilon) \operatorname{OPT}(V)$. □

Ainsi, à partir de maintenant, nous traiterons uniquement des instances correctement arrondies. Sans perte de généralité, supposons que toutes les coordonnées sont dans $[0, 2^N] \times [0, 2^N]$, où $N := \lceil \log L \rceil + 1$ et $L = \max_{v,w \in V} ||v - w||_2$. Nous partitionnons maintenant le carré $[0, 2^N] \times [0, 2^N]$ par une grille régulière : pour $i = 1, \ldots, N-1$ définissons $G_i := X_i \cup Y_i$, où

$$\begin{aligned} X_i &:= \left\{\left[\left(0, k2^{N-i}\right), \left(2^N, k2^{N-i}\right)\right] : k = 0, \ldots, 2^i - 1\right\}, \\ Y_i &:= \left\{\left[\left(j2^{N-i}, 0\right), \left(j2^{N-i}, 2^N\right)\right] : j = 0, \ldots, 2^i - 1\right\}. \end{aligned}$$

(On note $[(x, y), (x', y')]$ le segment de droite reliant (x, y) et (x', y').)

Plus précisément, étudions des **grilles décalées** : soient $a, b \in \{0, 2, \ldots, 2^N - 2\}$ des nombres pairs. Pour $i = 1, \ldots, N-1$, définissons $G_i^{(a,b)} := X_i^{(b)} \cup Y_i^{(a)}$, où

$$\begin{aligned} X_i^{(b)} := \Big\{\Big[\left(0, (b + k2^{N-i}) \bmod 2^N\right), \left(2^N, (b + k2^{N-i}) \bmod 2^N\right)\Big] : \\ k = 0, \ldots, 2^i - 1\Big\}, \end{aligned}$$

$$\begin{aligned} Y_i^{(a)} := \Big\{\Big[\left((a + j2^{N-i}) \bmod 2^N, 0\right), \left((a + j2^{N-i}) \bmod 2^N, 2^N\right)\Big] : \\ j = 0, \ldots, 2^i - 1\Big\}. \end{aligned}$$

(On note $x \bmod y$ le nombre unique z tel que $0 \leq z < y$ et $\frac{x-z}{y} \in \mathbb{Z}$.) Remarquons que $G_{N-1} = G_{N-1}^{(a,b)}$ ne dépend pas de a ou b.

On dit qu'une droite l est au **niveau** 1 si $l \in G_1^{(a,b)}$ et au niveau i si $l \in G_i^{(a,b)} \setminus G_{i-1}^{(a,b)}$ $(i = 2, \ldots, N-1)$. Voir la figure 21.3, où les droites représentées par les lignes les plus épaisses sont situées à des niveaux inférieurs. Les **régions** de la grille $G_i^{(a,b)}$ sont les ensembles

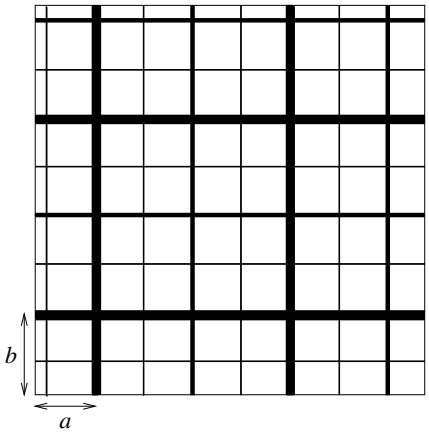


Figure 21.3.

$$\Big\{(x,y) \in [0,2^N) \times [0,2^N) \quad : \quad (x - a - j2^{N-i}) \bmod 2^N < 2^{N-i}, \\ (y - b - k2^{N-i}) \bmod 2^N < 2^{N-i}\Big\}$$

pour $j, k \in \{0, \ldots, 2^i - 1\}$. Pour $i < N - 1$, certaines des régions peuvent ne pas être connexes et être constituées de deux ou quatre rectangles. Comme toutes les droites sont définies par des coordonnées paires, aucune d'entre elles ne contient un point de notre instance du PVC EUCLIDIEN correctement arrondie. De plus, chaque région de G_{N-1} contient au plus un point, pour tout a, b.

Pour un polygone T et une droite l de G_{N-1}, nous noterons encore $cr(T, l)$ le nombre de fois où T croise l. La proposition suivante sera utile :

Proposition 21.9. *Pour un tour optimum T d'une instance correctement arrondie V du* PVC EUCLIDIEN, $\sum_{l \in G_{N-1}} cr(T, l) \leq \mathrm{OPT}(V)$.

Preuve. Considérons une arête de T de longueur s, de partie horizontale x et de partie verticale y. L'arête croise les droites de G_{N-1} au plus $\frac{x}{2} + 1 + \frac{y}{2} + 1$ fois. Puisque $x + y \leq \sqrt{2}s$ et que $s \geq 8$ (l'instance est correctement arrondie), l'arête croise les droites de G_{N-1} au plus $\frac{\sqrt{2}}{2}s + 2 \leq s$ fois. En sommant sur toutes les arêtes de T, on obtient l'inégalité voulue. □

Posons $C := 7 + \left\lceil \frac{36}{\epsilon} \right\rceil$ et $P := N \left\lceil \frac{6}{\epsilon} \right\rceil$. Pour chaque droite, définissons des **portails** : si $l = \left[\left(0, (b + k2^{N-i}) \bmod 2^N\right), \left(2^N, (b + k2^{N-i}) \bmod 2^N\right)\right]$ est une droite horizontale au niveau i, nous définissons l'ensemble de ses portails par

$$\left\{\left(a + \frac{h}{P}2^{N-i}, (b + k2^{N-i}) \bmod 2^N\right) : h = 0, \ldots, P2^i\right\}.$$

Les portails des droites verticales sont définis de manière analogue.

Définition 21.10. *Soit $V \subseteq [0, 2^N] \times [0, 2^N]$ une instance correctement arrondie du* PVC EUCLIDIEN. *Soient $a, b \in \{0, 2, \ldots, 2^N - 2\}$ donnés et considérons les grilles décalées, C, P et les portails définis comme ci-dessus. Un* **tour de Steiner** *est une chaîne fermée de segments de droites contenant V telle que son intersection avec chaque droite des grilles est un sous-ensemble de portails. Un tour de Steiner est* **léger** *si pour chaque i et pour chaque région de $G_i^{(a,b)}$, le tour croise chaque arête de la région au plus C fois.*

Remarquons que les tours de Steiner ne sont pas nécessairement des polygones (ils peuvent se croiser). Pour rendre un tour de Steiner léger, nous aurons à utiliser fréquemment le lemme de raccordement suivant :

Lemme 21.11. *Soit $V \subset \mathbb{R}^2$ une instance du* PVC EUCLIDIEN *et un tour T pour V. Soit l un segment de longueur s d'une droite ne contenant pas de point de V. Alors il existe un tour pour V dont la longueur dépasse celle de T d'au plus $6s$ et qui croise l au plus deux fois.*

Preuve. Pour rendre la démonstration plus claire, supposons que l est un segment de droite vertical. Supposons que T croise l exactement k fois, disons avec les arêtes $e_1, \ldots, e_k$. Supposons $k \geq 3$; sinon l'affirmation est triviale. Nous subdivisons chacune des arêtes $e_1, \ldots, e_k$ à l'aide de deux nouveaux sommets sans augmenter la longueur du tour. Autrement dit, nous remplaçons e_i par une chaîne de longueur 3 avec deux nouveaux sommets internes $p_i, q_i \in \mathbb{R}^2$ très proches de l et tels que p_i soit à gauche de l et q_i soit à droite de l ($i = 1, \ldots, k$). Notons T' le tour obtenu.

Considérons $t := \lfloor \frac{k-1}{2} \rfloor$ (alors $k - 2 \leq 2t \leq k - 1$) et soit T'' le tour déduit de T' par suppression des arêtes $(p_1, q_1), \ldots, (p_{2t}, q_{2t})$.

Soit P l'union d'un tour de longueur minimum contenant les sommets $p_1, \ldots, p_k$ et d'un couplage parfait de coût minimum des sommets $p_1, \ldots, p_{2t}$. Similairement, soit Q l'union d'un tour de longueur minimum passant par les sommets $q_1, \ldots, q_k$ et d'un couplage parfait de coût minimum des sommets $q_1, \ldots, q_{2t}$. La longueur totale des arêtes est au plus égale à $3s$ dans P et dans Q.

Alors $T'' + P + Q$ croise l au plus $k - 2t \leq 2$ fois et est connexe et eulérien. Nous procédons maintenant comme au lemme 21.3. D'après le théorème d'Euler 2.24, il existe un parcours eulérien dans $T'' + P + Q$. On peut alors le convertir en un tour pour V, en raccourcissant certaines chaînes, sans augmenter la longueur ou le nombre de croisements avec l. □

L'algorithme présenté est fondé principalement sur le théorème suivant :

Théorème 21.12. (Arora [1998]) *Soit $V \subseteq [0, 2^N] \times [0, 2^N]$ une instance correctement arrondie du* PVC EUCLIDIEN. *Si a et b sont choisis aléatoirement hors de $\{0, 2, \ldots, 2^N - 2\}$, alors, avec une probabilité supérieure ou égale à $\frac{1}{2}$, il existe un tour de Steiner léger dont la longueur est au plus égale à $(1 + \epsilon)\,\mathrm{OPT}(V)$.*

Preuve. Soit T un tour optimum pour V. Nous introduisons des points de Steiner à chaque fois que le tour croise une droite.

Nous déplaçons maintenant tous les points de Steiner jusqu'aux portails. Le portail le plus proche d'un point de Steiner sur une droite située au niveau i peut être éloigné d'au plus $\frac{2^{N-i-1}}{P}$. Puisqu'une droite l est au niveau i avec une probabilité $p(l,i) := \begin{cases} 2^{i-N} & \text{si } i > 1 \\ 2^{2-N} & \text{si } i = 1 \end{cases}$, l'espérance d'augmentation de la longueur du tour total, lorsque l'on déplace les points de Steiner de l jusqu'aux portails, est au plus égale à

$$\sum_{i=1}^{N-1} p(l,i) cr(T,l) 2\frac{2^{N-i-1}}{P} \;=\; N\frac{cr(T,l)}{P}.$$

Nous modifions maintenant le tour de Steiner de telle manière qu'il devienne léger. Considérons la procédure suivante :

For $i := N-1$ **down to** 1 **do** :
Appliquer le lemme de raccordement 21.11 à chaque segment d'une droite horizontale de $G_i^{(a,b)}$, c.-à-d. pour $j,k \in \{0,\ldots,2^i-1\}$ chaque droite entre $\big((a+j2^{N-i}) \bmod 2^N, (b+k2^{N-i}) \bmod 2^N\big)$ et $\big((a+(j+1)2^{N-i}) \bmod 2^N, (b+k2^{N-i}) \bmod 2^N\big)$, qui est croisée plus de $C-4$ fois.
Appliquer le lemme de raccordement 21.11 à chaque segment d'une droite verticale de $G_i^{(a,b)}$, c.-à-d. pour $j,k \in \{0,\ldots,2^i-1\}$ chaque droite entre $\big((a+j2^{N-i}) \bmod 2^N, (b+k2^{N-i}) \bmod 2^N\big)$ et $\big((a+j2^{N-i}) \bmod 2^N, (b+(k+1)2^{N-i}) \bmod 2^N\big)$, qui est croisée plus de $C-4$ fois.

Nous devons faire deux remarques. Un segment d'une droite horizontale ou verticale peut être constitué de deux parties séparées. Dans ce cas, le lemme de raccordement est appliqué à chaque partie et donc le nombre total de croisements peut après être égal à 4.

De plus, observons que l'application du lemme de raccordement à un segment d'une droite verticale l à l'itération i peut introduire de nouveaux croisements (points de Steiner) sur un segment de droite horizontale qui a une extrémité dans l. Ces nouveaux croisements sont situés sur des portails et ne seront plus considérés aux itérations suivantes de la procédure précédente, car ils sont sur des droites de niveau plus élevé.

Pour chaque droite l, le nombre d'applications du lemme de raccordement à l est inférieur ou égal à $\frac{cr(T,l)}{C-7}$, puisqu'à chaque fois le nombre de croisements diminue d'au moins $C-7$ (au moins $C-3$ croisements sont remplacés par au plus 4). Pour une droite l, notons $c(l,i,a,b)$ le nombre total de fois où le lemme de raccordement est appliqué à l à l'itération i de la procédure précédente. Remarquons que $c(l,i,a,b)$ ne dépend pas du niveau de l tant qu'il est inférieur ou égal à i.

Alors l'augmentation totale de la longueur du tour, due aux applications du lemme de raccordement à la droite l, est égale à $\sum_{i \geq level(l)} c(l,i,a,b) \cdot 6 \cdot 2^{N-i}$. De plus, $\sum_{i \geq level(l)} c(l,i,a,b) \leq \frac{cr(T,l)}{C-7}$.

Comme l est au niveau j avec une probabilité $p(l,j)$, l'espérance d'augmentation totale de la longueur du tour par la procédure précédente est inférieure ou égale à

$$\begin{aligned}\sum_{j=1}^{N-1} p(l,j) \sum_{i \geq j} c(l,i,a,b) \cdot 6 \cdot 2^{N-i} &= 6 \sum_{i=1}^{N-1} c(l,i,a,b) 2^{N-i} \sum_{j=1}^{i} p(l,j) \\ &\leq \frac{12 cr(T,l)}{C-7}.\end{aligned}$$

Après l'application de la procédure précédente, chaque segment de droite (et ainsi chaque arête d'une région) est croisé par le tour au plus $C-4$ fois, sans compter les nouveaux croisements induits par la procédure (voir la remarque précédente). Ces croisements supplémentaires sont tous situés à l'une des extrémités des segments de droite. Mais si un tour passe trois fois ou plus par le même point, deux des croisements peuvent être supprimés sans que cela augmente la longueur du tour ou que cela induise des croisements supplémentaires. (En supprimant deux arêtes parmi trois arêtes parallèles d'un graphe eulérien connexe, nous obtenons encore un graphe eulérien connexe.) Nous avons donc au plus quatre croisements supplémentaires pour chaque arête de chaque région (au plus deux pour chaque extrémité) et le tour est effectivement léger.

D'après la proposition 21.9, l'espérance d'augmentation de la longueur du tour est donc inférieure ou égale à

$$\sum_{l \in G_{N-1}} N \frac{cr(T,l)}{P} + \sum_{l \in G_{N-1}} \frac{12 cr(T,l)}{C-7} \leq \mathrm{OPT}(V) \left(\frac{N}{P} + \frac{12}{C-7} \right) \leq \mathrm{OPT}(V) \frac{\epsilon}{2}.$$

Ainsi la probabilité que l'augmentation de la longueur du tour soit inférieure ou égale à $\mathrm{OPT}(V)\epsilon$ est supérieure ou égale à $\frac{1}{2}$. □

À l'aide de ce théorème, nous pouvons finalement décrire l'ALGORITHME D'ARORA. L'idée est d'énumérer tous les tours de Steiner légers, en utilisant la programmation dynamique. Un **sous-problème** est constitué d'une région r d'une grille $G_i^{(a,b)}$ avec $1 \leq i \leq N-1$, d'un ensemble A de cardinalité paire, dont chaque élément est assigné à un portail sur l'une des arêtes de r (de telle façon qu'il n'y ait pas plus de C éléments assignés à une arête) et d'un couplage parfait M du graphe complet sur A. Donc, pour chaque région, nous avons au plus $(P+2)^{4C}(4C)!$ sous-problèmes (à l'ordre près des éléments de A). Une solution d'un tel sous-problème est un ensemble de $|M|$ chaînes $\{P_e : e \in M\}$ qui correspond à l'intersection d'un tour de Steiner léger pour V avec la région r, tel que $P_{\{v,w\}}$ ait pour extrémités v et w et que chaque point de $V \cap r$ appartienne à une seule des chaînes. Une solution est optimale si la longueur totale des chaînes est la plus petite possible.

ALGORITHME D'ARORA	
Input	Une instance correctement arrondie $V \subseteq [0,2^N] \times [0,2^N]$ du PVC EUCLIDIEN. Un nombre $0 < \epsilon < 1$.
Output	Un tour optimal à un facteur $(1+\epsilon)$ près.

① Choisir a et b aléatoirement hors de $\{0, 2, \ldots, 2^N - 2\}$.
Poser $R_0 := \big\{\big([0, 2^N] \times [0, 2^N], V\big)\big\}$.

② **For** $i := 1$ **to** $N - 1$ **do** :
Construire $G_i^{(a,b)}$. Poser $R_i := \emptyset$.
For chaque $(r, V_r) \in R_{i-1}$ tel que $|V_r| \geq 2$ **do** :
Construire les quatre régions r_1, r_2, r_3, r_4 de $G_i^{(a,b)}$ telles que $r_1 \cup r_2 \cup r_3 \cup r_4 = r$ et ajouter $(r_j, V_r \cap r_j)$ à R_i $(j = 1, 2, 3, 4)$.

③ **For** $i := N - 1$ **down to** 1 **do** :
For chaque région $r \in R_i$ **do** : résoudre tous ses sous-problèmes de manière optimale.
If $|V_r| \leq 1$ **then** cela est immédiat,
else on utilise les solutions optimales déjà calculées pour les sous-problèmes des quatre sous-régions.

④ Calculer un tour de Steiner léger pour V optimum en utilisant les solutions optimales des sous-problèmes des quatre sous-régions.
Supprimer les points de Steiner pour obtenir un tour qui n'est pas plus long.

Théorème 21.13. *L'*ALGORITHME D'ARORA *trouve un tour qui est, avec une probabilité supérieure ou égale à $\frac{1}{2}$, de longueur inférieure ou égale à $(1+\epsilon)\,\mathrm{OPT}(V)$. Le temps de calcul est $O(n(\log n)^c)$ pour une constante c (dépendant linéairement de $\frac{1}{\epsilon}$).*

Preuve. L'algorithme choisit a et b aléatoirement et calcule ensuite un tour de Steiner léger optimum. D'après le théorème 21.12, ce tour est de longueur inférieure ou égale à $(1 + \epsilon)\,\mathrm{OPT}(V)$ avec une probabilité supérieure ou égale à $\frac{1}{2}$. La suppression finale des points de Steiner peut uniquement raccourcir le tour.

Afin d'estimer le temps de calcul, considérons l'arbre des régions : la racine est la région définie par R_0 et chaque région $r \in R_i$ a zéro ou quatre enfants (sous-régions de R_{i+1}). Soit S l'ensemble des sommets de cet arbre qui ont quatre enfants qui sont tous des feuilles. Comme les intérieurs de ces régions sont disjoints et que chacun contient au moins deux points de V, nous avons $|S| \leq \frac{n}{2}$. Comme chaque sommet de l'arbre est soit une feuille soit un prédécesseur d'au moins un sommet de S, on a au plus $N\frac{n}{2}$ sommets qui ne sont pas des feuilles et ainsi au plus $\frac{5}{2}Nn$ sommets au total.

Pour chaque région, il apparaît au plus $(P+2)^{4C}(4C)!$ sous-problèmes. On peut résoudre directement en temps $O(C)$ les sous-problèmes correspondant aux régions qui contiennent au plus un point. Pour les autres sous-problèmes, on considère tous les multi-ensembles possibles de portails sur les quatre arêtes entre les sous-régions et tous les ordres possibles dans lesquels on peut traverser les portails. Il y a ainsi au plus $(P + 2)^{4C}(8C)!$ possibilités et elles peuvent alors être évaluées en temps constant en utilisant les solutions stockées des sous-problèmes.

Donc, pour tous les sous-problèmes d'une région, le temps de calcul est $O\big((P+2)^{8C}(4C)!\ (8C)!\big)$. Observons que cela est également vrai pour les régions non

connexes : puisque le tour ne peut pas passer d'une composante connexe d'une région à une autre, le problème ne peut que devenir plus facile.

Puisque l'on ne considère qu'au plus $\frac{5}{2}Nn$ régions, que $N = O\left(\log \frac{n}{\epsilon}\right)$ (l'instance est correctement arrondie), $C = O\left(\frac{1}{\epsilon}\right)$ et que $P = O\left(\frac{N}{\epsilon}\right)$, on obtient un temps de calcul total

$$O\left(n \log \frac{n}{\epsilon}(P+2)^{8C}(8C)^{12C}\right) = O\left(n(\log n)^{O\left(\frac{1}{\epsilon}\right)} O\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon}\right)}\right). \qquad \square$$

Bien entendu, l'ALGORITHME D'ARORA peut facilement être dérandomisé en essayant toutes les valeurs possibles pour a et b. Cela ajoute un facteur $O\left(\frac{n^2}{\epsilon^2}\right)$ au temps de calcul. Nous en concluons :

Corollaire 21.14. *Il existe un schéma d'approximation pour le* PVC EUCLIDIEN. *Pour tout $\epsilon > 0$ fixé, on peut déterminer une solution approchée à un facteur $(1+\epsilon)$ près en temps $O\left(n^3(\log n)^c\right)$ pour une certaine constante c.* $\square$

Rao et Smith [1998] ont réduit le temps de calcul à $O(n \log n)$ pour tout $\epsilon > 0$ fixé. Cependant, les constantes mises en jeu sont encore assez grandes pour des valeurs raisonnables de ϵ et l'intérêt pratique de leur méthode paraît donc limité. Klein [2005] a trouvé un schéma d'approximation dans le cas où l'instance est la fermeture métrique d'un graphe planaire avec des poids non négatifs sur les arêtes.

21.3 Méthodes locales

En général, la technique la plus efficace en pratique, pour obtenir de bonnes solutions à des instances du PVC, est la recherche locale. L'idée est la suivante. Nous partons d'un tour trouvé avec une heuristique. Nous essayons alors d'améliorer notre solution à l'aide de certaines modifications «locales». Par exemple, nous pourrions couper le tour en deux morceaux en supprimant deux arêtes et ensuite joindre les morceaux pour former un tour différent.

La recherche locale est plus un principe algorithmique qu'un algorithme. En particulier, deux décisions doivent être prises à l'avance :

- quelles sont les modifications autorisées, c.-à-d. comment est défini le voisinage d'une solution ?
- quand modifions-nous réellement la solution (une possibilité est de n'autoriser que les améliorations) ?

Pour donner un exemple concret, nous décrivons l'ALGORITHME k-OPT bien connu pour le PVC. Soit $k \geq 2$ un entier fixé.

ALGORITHME k-OPT	
Input	Une instance (K_n, c) du PVC.
Output	Un tour.

① Soit T un tour.

② Soit $\mathcal{S}$ la famille des sous-ensembles à k éléments de $E(T)$.

③ **For** tout $S \in \mathcal{S}$ et tous les tours T' tels que $E(T') \supseteq E(T) \setminus S$ **do** :
If $c(E(T')) < c(E(T))$ **then** poser $T := T'$ et **go to** ②.

④ **Stop**.

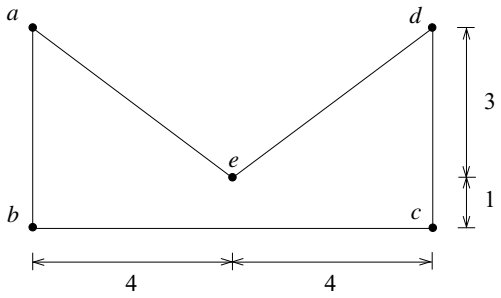


Figure 21.4.

Un tour est dit ***k*-opt** s'il ne peut pas être amélioré à l'aide de l'ALGORITHME k-OPT. Pour tout k fixé, il existe des instances du PVC et des tours k-opt qui ne sont pas $(k+1)$-opt. Par exemple, le tour représenté à la figure 21.4 est 2-opt mais pas 3-opt (respectivement à la distance euclidienne). Il peut être amélioré en échangeant trois arêtes (le tour (a, b, e, c, d, a) est optimum).

Le tour représenté sur la droite de la figure 21.5 est 3-opt relativement aux poids indiqués sur le graphe situé à gauche. Les arêtes qui ne sont pas représentées ont un poids égal à 4. Cependant, on peut obtenir immédiatement la solution optimale en échangeant quatre arêtes. Remarquons que l'inégalité triangulaire est vérifiée.

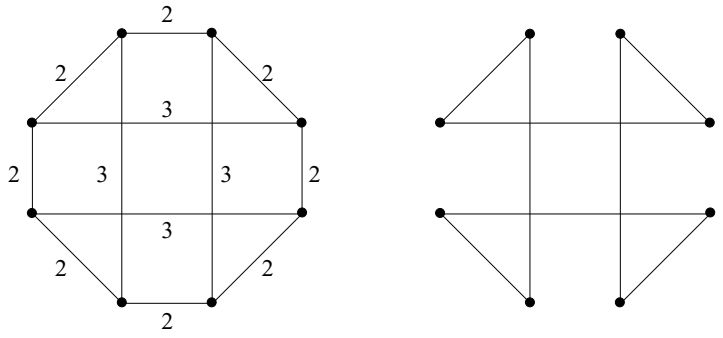


Figure 21.5.

En fait, la situation est bien pire : un tour obtenu à l'aide de l'ALGORITHME k-OPT, pour une instance à n villes, peut être jusqu'à $\frac{1}{4}n^{\frac{1}{2k}}$ fois plus long que le tour optimum (pour tout k et un nombre infini de n) ; la longueur d'un tour 2-opt n'est jamais supérieure à $4\sqrt{n}$ fois la longueur optimale. Cependant, le temps de calcul dans le pire des cas de l'ALGORITHME k-OPT est exponentiel pour tout k et cela est vrai même pour l'algorithme 2-OPT sur des instances euclidiennes. Ces résultats sont dus à Chandra, Karloff et Tovey [1999] et Englert, Röglin et Vöcking [2007].

Une autre question concerne la manière de choisir k à l'avance. Bien entendu, les instances (K_n, c) sont résolues optimalement par l'ALGORITHME k-OPT si $k = n$, mais le temps de calcul augmente de manière exponentielle relativement à k. Prendre $k = 3$ est souvent un bon choix. Lin et Kernighan [1973] ont proposé une heuristique efficace dans laquelle k n'est pas fixé, mais plutôt déterminé par l'algorithme. Leur idée est fondée sur le concept suivant :

Définition 21.15. *Étant donné une instance (K_n, c) du* PVC *et un tour T. Une* **chaîne alternée** *est une suite de sommets (villes) $P = (x_0, x_1, \ldots, x_{2m})$ telle que $(x_i, x_{i+1}) \neq (x_j, x_{j+1})$ pour tout $0 \leq i < j < 2m$ et telle que pour $i = 0, \ldots, 2m-1$ on ait $(x_i, x_{i+1}) \in E(T)$ si et seulement si i est pair. P est* **fermée** *si de plus $x_0 = x_{2m}$.*

Le **gain** *de P est défini par*

$$g(P) := \sum_{i=0}^{m-1} (c((x_{2i}, x_{2i+1})) - c((x_{2i+1}, x_{2i+2}))).$$

P est dite **propre** *si $g((x_0, \ldots, x_{2i})) > 0$ pour tout $i \in \{1, \ldots, m\}$. Nous utilisons l'abréviation $E(P) = \{(x_i, x_{i+1}) : i = 0, \ldots, 2m-1\}$.*

Remarquons que les sommets peuvent être traversés plus d'une fois dans une chaîne alternée. Dans l'exemple représenté à la figure 21.4, (a, e, b, c, e, d, a) est une chaîne alternée fermée propre. Étant donné un tour T, nous nous intéressons bien entendu aux chaînes alternées fermées P pour lesquelles $E(T) \triangle E(P)$ définit encore un tour.

Lemme 21.16. (Lin et Kernighan [1973]) *S'il existe une chaîne alternée fermée P telle que $g(P) > 0$, alors :*

(a) $c(E(T) \triangle E(P)) = c(E(T)) - g(P)$.

(b) *Il existe une chaîne alternée fermée propre Q telle que $E(Q) = E(P)$.*

Preuve. La partie (a) est une conséquence directe de la définition. Pour prouver la partie (b), considérons $P = (x_0, x_1, \ldots, x_{2m})$ et notons k le plus grand indice tel que $g((x_0, \ldots, x_{2k}))$ est minimum. Nous affirmons que $Q := (x_{2k}, x_{2k+1}, \ldots, x_{2m-1}, x_0, x_1, \ldots, x_{2k})$ est propre. Pour $i = k+1, \ldots, m$, nous avons

$$g((x_{2k}, x_{2k+1}, \ldots, x_{2i})) = g((x_0, x_1, \ldots, x_{2i})) - g((x_0, x_1, \ldots, x_{2k})) > 0$$

par définition de k. Pour $i = 1, \ldots, k$, nous avons

$$\begin{aligned} & g((x_{2k}, x_{2k+1}, \ldots, x_{2m-1}, x_0, x_1, \ldots, x_{2i})) \\ = \; & g((x_{2k}, x_{2k+1}, \ldots, x_{2m})) + g((x_0, x_1, \ldots, x_{2i})) \\ \geq \; & g((x_{2k}, x_{2k+1}, \ldots, x_{2m})) + g((x_0, x_1, \ldots, x_{2k})) \\ = \; & g(P) \; > \; 0, \end{aligned}$$

de nouveau par définition de k. Donc Q est bien une chaîne alternée fermée propre. □

Nous passons maintenant à la description de l'algorithme. Étant donné un tour T, l'algorithme recherche une chaîne alternée fermée propre P et itère ensuite avec $(V(T), E(T) \triangle E(P))$. À chaque itération, il vérifie de manière exhaustive toutes les possibilités jusqu'à ce qu'une chaîne alternée fermée propre soit trouvée, ou jusqu'à ce que l'un des deux paramètres p_1 et p_2 l'empêche de continuer ainsi. Voir également la figure 21.6 pour une illustration.

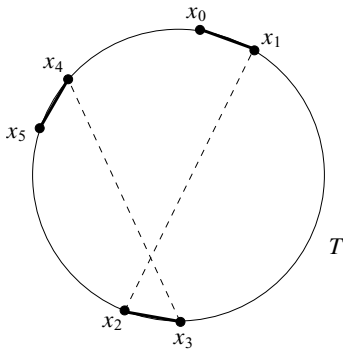


Figure 21.6.

ALGORITHME DE LIN-KERNIGHAN	
Input	Une instance (K_n, c) du PVC. Deux paramètres $p_1 \in \mathbb{N}$ (profondeur du *backtracking*) et $p_2 \in \mathbb{N}$ (profondeur de non-réalisabilité).
Output	Un tour T.

① Soit T un tour.

② Poser $X_0 := V(K_n)$, $i := 0$ et $g^* := 0$.

③ **If** $X_i = \emptyset$ et $g^* > 0$ **then** :
Poser $T := (V(T), E(T) \triangle E(P^*))$ et **go to** ②.
If $X_i = \emptyset$ et $g^* = 0$ **then** :
Poser $i := \min(i-1, p_1)$. **If** $i < 0$ **then stop, else go to** ③.

④ Choisir $x_i \in X_i$ et poser $X_i := X_i \setminus \{x_i\}$.
If i est impair, $i \geq 3$, $(V(T), E(T) \triangle E((x_0, x_1, \ldots, x_{i-1}, x_i, x_0)))$ est un tour et $g((x_0, x_1, \ldots, x_{i-1}, x_i, x_0)) > g^*$ **then** :
Poser $P^* := (x_0, x_1, \ldots, x_{i-1}, x_i, x_0)$ et $g^* := g(P^*)$.

⑤ **If** i est impair **then** :
Poser $X_{i+1} := \{\in V(K_n) \setminus \{x_0, x_i\} :$
$(x_i, x) \notin E(T) \cup E((x_0, x_1, \ldots, x_{i-1}))$,
$g((x_0, x_1, \ldots, x_{i-1}, x_i, x)) > g^*\}$.
If i est pair et $i \leq p_2$ **then** :
Poser $X_{i+1} := \{x \in V(K_n) : (x_i, x) \in E(T) \setminus E((x_0, x_1, \ldots, x_i))\}$.
If i est pair et $i > p_2$ **then** :
Poser $X_{i+1} := \{x \in V(K_n) : (x_i, x) \in E(T) \setminus E((x_0, x_1, \ldots, x_i))$,
$\{x, x_0\} \notin E(T) \cup E((x_0, x_1, \ldots, x_i))$,
$(V(T), E(T) \triangle E((x_0, x_1, \ldots, x_i, x, x_0)))$ est un tour$\}$.

Poser $i := i + 1$. **Go to** ③.

Lin et Kernighan ont proposé les paramètres $p_1 = 5$ et $p_2 = 2$. Ce sont les plus petites valeurs garantissant que l'algorithme trouve un échange favorable de trois arêtes :

Théorème 21.17. (Lin et Kernighan [1973])
(a) *Pour $p_1 = \infty$ et $p_2 = \infty$, l'*ALGORITHME DE LIN-KERNIGHAN *trouve une chaîne alternée fermée propre P telle que $(V(T), E(T) \triangle E(P))$ soit un tour, s'il en existe une.*
(b) *Pour $p_1 = 5$ et $p_2 = 2$, l'*ALGORITHME DE LIN-KERNIGHAN *retourne un tour qui est 3-opt.*

Preuve. Soit T le tour avec lequel l'algorithme termine. Alors g^* doit être égal à zéro depuis le dernier changement de tour. Cela implique que, dans le cas où $p_1 = p_2 = \infty$, l'algorithme a énuméré entièrement toutes les chaînes alternées propres. En particulier, (a) est vérifié.

Dans le cas où $p_1 = 5$ et $p_2 = 2$, l'algorithme a au moins énuméré toutes les chaînes alternées fermées propres de longueur égale à 4 ou 6. Supposons qu'il existe un échange favorable de deux ou trois arêtes résultant en un tour T'. Alors les arêtes $E(T) \triangle E(T')$ forment une chaîne alternée fermée P avec au plus six arêtes et telle que $g(P) > 0$. D'après le lemme 21.16, sans perte de généralité P est propre et l'algorithme l'aurait trouvé. Cela prouve (b). □

Remarquons que cette procédure ne peut pas trouver un échange «non séquentiel» tel que l'échange de quatre arêtes montré à la figure 21.5. Dans cet exemple, le tour ne peut pas être amélioré à l'aide de l'ALGORITHME DE LIN-KERNIGHAN, mais un échange de quatre arêtes (non séquentiel) fournirait la solution optimale.

On peut donc proposer l'amélioration suivante de l'ALGORITHME DE LIN-KERNIGHAN. Si l'algorithme s'arrête, on essaye (à l'aide d'une heuristique) de

trouver un échange de quatre arêtes non séquentiel et favorable. S'il en existe un, on continue avec le nouveau tour, sinon on abandonne.

L'ALGORITHME DE LIN-KERNIGHAN est bien plus efficace que, par exemple, l'ALGORITHME 3-OPT. Tout en étant au moins aussi bonne (et habituellement bien meilleure), l'espérance du temps de calcul (avec $p_1 = 5$ et $p_2 = 2$) se compare favorablement : Lin et Kernighan ont proposé un temps de calcul empirique d'environ $O(n^{2.2})$. Cependant, il semble peu probable que le temps de calcul dans le pire des cas soit polynomial ; pour une formulation précise de cette affirmation (et une preuve), voir l'exercice 11 (Papadimitriou [1992]).

Presque toutes les heuristiques de recherche locale, utilisées en pratique pour le PVC, sont fondées sur cet algorithme. Bien que le comportement de l'ALGORITHME DE LIN-KERNIGHAN soit plus mauvais que l'ALGORITHME DE CHRISTOFIDES dans le pire des cas, il fournit généralement de bien meilleures solutions, habituellement à quelques pour cent près de l'optimum. Pour une variante très efficace, voir Applegate, Cook et Rohe [2003].

D'après l'exercice 14 du chapitre 9, il n'existe pas d'algorithme de recherche locale pour le PVC qui ait une complexité polynomiale par itération et trouve toujours une solution optimale, à moins que *P = NP* (ici une itération correspond au calcul entre deux modifications du tour courant). Nous montrons maintenant que l'on ne peut même pas décider si un tour donné est optimum. Pour cela, nous considérons d'abord la restriction suivante du PROBLÈME DU CYCLE HAMILTONIEN :

CYCLE HAMILTONIEN RESTREINT	
Instance	Un graphe non orienté G et une chaîne hamiltonienne dans G.
Question	G contient-il un cycle hamiltonien ?

Lemme 21.18. *Le* PROBLÈME DU CYCLE HAMILTONIEN RESTREINT *est NP-complet.*

Preuve. Étant donné une instance G du PROBLÈME DU CYCLE HAMILTONIEN (qui est *NP*-complet, voir théorème 15.25), nous construisons une instance équivalente du problème du CYCLE HAMILTONIEN RESTREINT.

Supposons que $V(G) = \{1, \ldots, n\}$. Nous considérons n copies du «graphe diamant» représenté à la figure 21.7 et nous les relions verticalement avec des arêtes (S_i, N_{i+1}) $(i = 1, \ldots, n-1)$.

Il est évident que le graphe ainsi obtenu contient une chaîne hamiltonienne de N_1 à S_n. Nous ajoutons maintenant des arêtes (W_i, E_j) et (W_j, E_i) pour toute arête $(i, j) \in E(G)$. Notons H le graphe obtenu. Il est évident que tout cycle hamiltonien de G induit un cycle hamiltonien de H.

De plus, un cycle hamiltonien de H doit traverser tous les sous-graphes diamants de la même façon : soit tous de E_i à W_i, soit tous de S_i à N_i. Mais cette dernière situation est impossible, donc H est hamiltonien si et seulement si G l'est. □

Théorème 21.19. (Papadimitriou et Steiglitz [1977]) *Le problème de décider si un tour donné est optimum pour une instance donnée du* PVC MÉTRIQUE *est coNP-complet.*

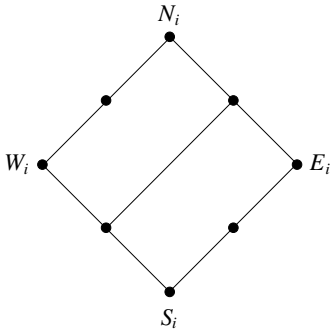


Figure 21.7.

Preuve. L'appartenance à *coNP* est claire, puisqu'un tour optimum sert de certificat pour la sous-optimalité.

Nous allons maintenant réduire le problème du CYCLE HAMILTONIEN RESTREINT au complémentaire de notre problème. Étant donné un graphe G et une chaîne hamiltonienne P dans G, nous vérifions d'abord si les extrémités de P sont reliées par une arête. Si c'est le cas, nous avons terminé. Sinon nous définissons

$$c_{ij} := \begin{cases} 1 & \text{si } (i,j) \in E(G) \\ 2 & \text{si } (i,j) \notin E(G). \end{cases}$$

L'inégalité triangulaire est bien entendu vérifiée. De plus, P définit un tour de coût $n+1$, qui est optimum si et seulement s'il n'existe pas de cycle hamiltonien dans G. □

Corollaire 21.20. *À moins que $P = NP$, aucun algorithme de recherche locale pour le* PVC*, ayant une complexité polynomiale par itération, ne peut être exact.*

Preuve. Un algorithme de recherche locale exact nécessite de décider si le tour initial est optimum. □

La recherche locale s'applique bien entendu à beaucoup d'autres problèmes d'optimisation combinatoire. L'ALGORITHME DU SIMPLEXE peut être aussi considéré comme un algorithme de recherche locale. Bien que les algorithmes de recherche locale se révèlent très efficaces en pratique, on ne connaît presque pas de preuves théoriques de leur efficacité, sauf dans quelques cas particuliers (voir, par exemple, l'exercice 10 au chapitre 16 et les paragraphes 22.6 et 22.8). Pour de nombreux problèmes *NP*-difficiles (incluant ceux présentés dans ce paragraphe), on ne sait même pas si l'on peut calculer un optimum local en temps polynomial ; voir l'exercice 11. Le livre édité par Aarts et Lenstra [1997] contient plus d'exemples d'heuristiques de recherche locale. Michiels, Aarts et Korst [2007] décrivent des résultats plus théoriques sur la recherche locale.

21.4 Polytope du voyageur de commerce

Dantzig, Fulkerson et Johnson [1954] ont été les premiers à résoudre optimalement une instance du PVC de taille respectable. Ils commencèrent par résoudre une relaxation linéaire d'une formulation par programmation linéaire convenable et ensuite ajoutèrent successivement des plans coupants. Cela a été le point de départ de l'analyse du polytope du voyageur de commerce :

Définition 21.21. *Pour $n \geq 3$, nous notons $Q(n)$ le* **polytope du voyageur de commerce***, c.-à-d. l'enveloppe convexe des vecteurs d'incidence des tours du graphe complet K_n.*

Bien que l'on ne connaisse pas de description complète du polytope du voyageur de commerce, il existe plusieurs résultats intéressants, certains d'entre eux sont également utiles pour les calculs pratiques. Puisque $\sum_{e \in \delta(v)} x_e = 2$ pour tout $v \in V(K_n)$ et tout $x \in Q(n)$, la dimension de $Q(n)$ est inférieure ou égale à $|E(K_n)| - |V(K_n)| = \binom{n}{2} - n = \frac{n(n-3)}{2}$. Afin de prouver qu'en fait $\dim(Q(n)) = \frac{n(n-3)}{2}$, nous avons besoin du lemme de théorie des graphes suivant :

Lemme 21.22. *Pour tout $k \geq 1$:*
(a) *L'ensemble des arêtes de K_{2k+1} peut être partitionné en k tours.*
(b) *L'ensemble des arêtes de K_{2k} peut être partitionné en $k-1$ tours et un couplage parfait.*

Preuve. (a) : supposons que les sommets soient numérotés $0, \ldots, 2k-1, x$. Considérons les tours

$$\begin{aligned} T_i \;=\; & (x, i, i+1, i-1, i+2, i-2, i+3, \ldots, \\ & i-k+2, i+k-1, i-k+1, i+k, x) \end{aligned}$$

pour $i = 0, \ldots, k-1$ (on considère tout modulo $2k$). Voir la figure 21.8 pour une illustration. Puisque $|E(K_{2k+1})| = k(2k+1)$, il suffit de montrer que ces tours sont arête-disjoints. Cela est évident pour les arêtes incidentes à x. De plus, pour $(a, b) \in E(T_i)$ tel que $a, b \neq x$, on a $a+b \in \{2i, 2i+1\}$, comme on peut facilement le vérifier.

(b) : supposons que les sommets soient numérotés $0, \ldots, 2k-2, x$. Considérons les tours

$$\begin{aligned} T_i \;=\; & (x, i, i+1, i-1, i+2, i-2, i+3, \ldots, \\ & i+k-2, i-k+2, i+k-1, i-k+1, x) \end{aligned}$$

pour $i = 0, \ldots, k-2$ (on considère tout modulo $2k-1$). Les mêmes arguments que précédemment montre que les tours sont arête-disjoints. Après les avoir supprimés, le graphe restant est 1-régulier et fournit donc un couplage parfait. □

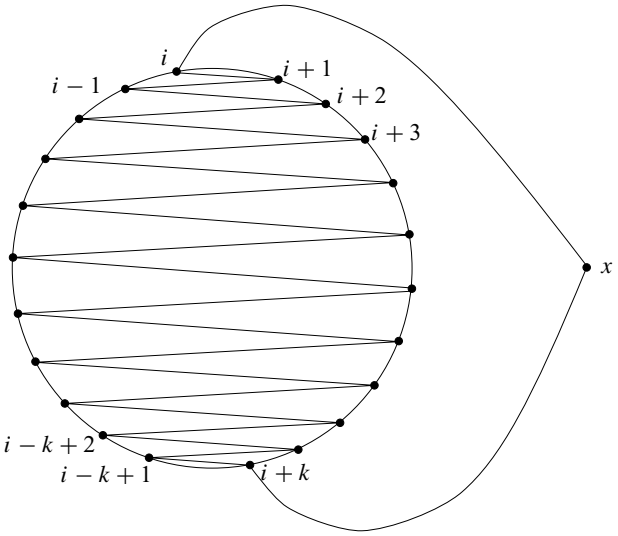


Figure 21.8.

Théorème 21.23. (Grötschel et Padberg [1979])

$$\dim\left(Q(n)\right) \;=\; \frac{n(n-3)}{2}.$$

Preuve. Pour $n = 3$, l'affirmation est triviale. Soit $n \geq 4$, considérons un sommet arbitraire $v \in V(K_n)$.
Cas 1 : n est pair, donc $n = 2k + 2$ pour un entier $k \geq 1$. D'après le lemme 21.22(a), $K_n - v$ est l'union de k tours arête-disjoints $T_0, \ldots, T_{k-1}$. Soit maintenant T_{ij}, obtenu à partir de T_i en remplaçant la j-ième arête (a, b) par deux arêtes (a, v), (v, b) $(i = 0, \ldots, k-1\,;\, j = 1, \ldots, n-1)$. Considérons la matrice dont les lignes sont les vecteurs d'incidence de ces $k(n-1)$ tours. Alors les colonnes correspondant aux arêtes non incidentes à v forment une matrice carrée

$$\begin{pmatrix} A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ 0 & 0 & A & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}, \text{ où } A = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix}.$$

Puisque la matrice est non singulière, les vecteurs d'incidence des $k(n-1)$ tours sont linéairement indépendants, ce qui implique $\dim\left(Q(n)\right) \geq k(n-1) - 1 = \frac{n(n-3)}{2}$.
Cas 2 : n est impair, donc $n = 2k + 3$ pour un entier $k \geq 1$. D'après le lemme 21.22(b), $K_n - v$ est l'union de k tours et d'un couplage parfait M. À partir des tours, nous construisons $k(n-1)$ tours dans K_n comme en (a). Nous complétons

maintenant arbitrairement le couplage parfait M pour obtenir un tour T dans K_{n-1}. Pour chaque arête $e = (a, b)$ de M, nous remplaçons e dans T par les deux arêtes (a, v) et (v, b). De cette façon, nous obtenons $k+1$ tours supplémentaires. De même que ci-dessus, les vecteurs d'incidence des $k(n-1)+k+1 = kn+1$ tours sont linéairement indépendants, ce qui prouve $\dim(Q(n)) \geq kn+1-1 = \frac{n(n-3)}{2}$. □

Les points entiers de $Q(n)$, c.-à-d. les tours, peuvent être décrits de la façon suivante :

Proposition 21.24. *Les vecteurs d'incidence des tours de K_n sont exactement les vecteurs entiers x vérifiant*

$$0 \leq x_e \leq 1 \qquad (e \in E(K_n)); \tag{21.1}$$

$$\sum_{e \in \delta(v)} x_e = 2 \qquad (v \in V(K_n)); \tag{21.2}$$

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \qquad (\emptyset \neq X \subset V(K_n)). \tag{21.3}$$

Preuve. Évidemment, le vecteur d'incidence d'un tour vérifie ces contraintes. Tout vecteur entier vérifiant (21.1) et (21.2) est le vecteur d'incidence d'un 2-couplage simple parfait, c.-à-d. l'union de cycles sommet-disjoints couvrant tous les sommets. Les contraintes (21.3) suppriment les cycles ayant moins de n arêtes. □

Les contraintes (21.3) sont habituellement appelées les **inégalités de sous-tours** et le polytope défini par (21.1), (21.2), (21.3) est appelé le **polytope des sous-tours**. En général, le polytope des sous-tours n'est pas entier, comme l'instance représentée sur la figure 21.9 le montre (les arêtes non représentées sont de poids égal à 3) : le tour le plus court est de longueur 10, mais la solution optimale fractionnaire ($x_e = 1$ si e est de poids égal à 1 et $x_e = \frac{1}{2}$ si e est de poids égal à 2) est de poids total égal à 9.

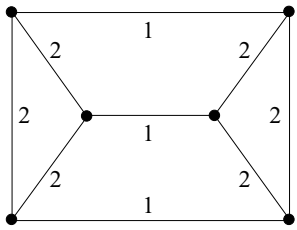


Figure 21.9.

Les descriptions suivantes du polytope des sous-tours sont équivalentes et se révéleront utiles :

Proposition 21.25. *Soit $V(K_n) = \{1, \dots, n\}$. Soit $x \in [0,1]^{E(K_n)}$ tel que $\sum_{e \in \delta(v)} x_e = 2$ pour tout $v \in V(K_n)$. Alors les affirmations suivantes sont équivalentes :*

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \qquad (\emptyset \neq X \subset V(K_n)); \tag{21.3}$$

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \qquad (\emptyset \neq X \subseteq V(K_n) \setminus \{1\}); \tag{21.4}$$

$$\sum_{e \in \delta(X)} x_e \geq 2 \qquad (\emptyset \neq X \subset V(K_n)). \tag{21.5}$$

Preuve. Pour tout $\emptyset \neq X \subset V(K_n)$, on a

$$\begin{aligned} \sum_{e \in \delta(V(K_n) \setminus X)} x_e = \sum_{e \in \delta(X)} x_e &= \sum_{v \in X} \sum_{e \in \delta(v)} x_e - 2 \sum_{e \in E(K_n[X])} x_e \\ &= 2|X| - 2 \sum_{e \in E(K_n[X])} x_e, \end{aligned}$$

ce qui implique l'équivalence de (21.3), (21.4) et (21.5). □

Corollaire 21.26. *Le* PROBLÈME DE SÉPARATION *pour les inégalités de sous-tours peut être résolu en temps polynomial.*

Preuve. En utilisant (21.5) et en considérant x comme des capacités associées aux arêtes, nous devons décider s'il existe une coupe dans (K_n, x) de capacité inférieure à 2. Ainsi le PROBLÈME DE SÉPARATION se réduit au problème de la recherche d'une coupe minimum dans un graphe non orienté avec des capacités non négatives. D'après le théorème 8.39, ce problème peut être résolu en temps $O(n^3)$. □

Puisqu'un tour est un 2-couplage simple parfait, l'enveloppe convexe de tous les 2-couplages simples parfaits contient le polytope du voyageur de commerce. Donc, d'après le théorème 12.3, nous avons :

Proposition 21.27. *Tout $x \in Q(n)$ vérifie les inégalités*

$$\sum_{e \in E(K_n[X]) \cup F} x_e \leq |X| + \frac{|F| - 1}{2} \quad \text{pour } X \subseteq V(K_n),\ F \subseteq \delta(X) \text{ et } |F| \text{ impair.} \tag{21.6}$$

Les contraintes (21.6) sont appelées **inégalités de 2-couplages**. Il est suffisant de considérer les inégalités (21.6) dans le cas où F est un couplage. Les autres inégalités de 2-couplages sont impliquées par celles-ci (exercice 13). Pour les inégalités de 2-couplages, le PROBLÈME DE SÉPARATION peut être résolu en temps polynomial, d'après le théorème 12.18. Donc, d'après la MÉTHODE DES ELLIPSOÏDES (théorème 4.21), on peut optimiser une fonction linéaire sur le polytope défini par (21.1), (21.2), (21.3) et (21.6) en temps polynomial (exercice 12). Les inégalités de 2-couplages se généralisent aux **inégalités de peignes**, illustrées par la figure 21.10 :

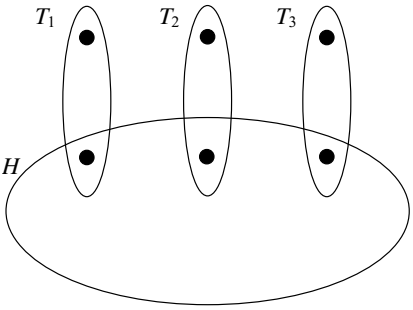


Figure 21.10.

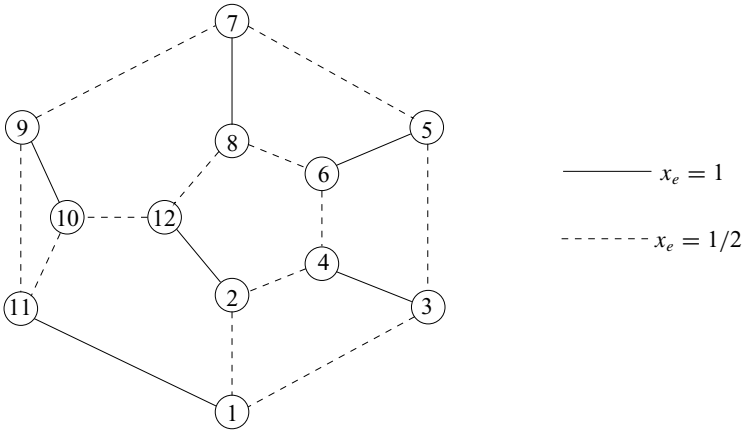


Figure 21.11.

Proposition 21.28. (Chvátal [1973], Grötschel et Padberg [1979]) *Soient* $T_1, \ldots, T_s \subseteq V(K_n)$, s *ensembles deux à deux disjoints,* $s \geq 3$ *impair et* $H \subseteq V(K_n)$ *tel que* $T_i \cap H \neq \emptyset$ *et* $T_i \setminus H \neq \emptyset$ *pour* $i = 1, \ldots, s$. *Alors tout* $x \in Q(n)$ *vérifie*

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{s} \sum_{e \in \delta(T_i)} x_e \geq 3s + 1. \tag{21.7}$$

Preuve. Soit x le vecteur d'incidence d'un tour. Pour tout $i \in \{1, \ldots, s\}$, on a

$$\sum_{e \in \delta(T_i)} x_e + \sum_{e \in \delta(H) \cap E(K_n[T_i])} x_e \geq 3,$$

puisque le tour doit entrer et sortir de $T_i \setminus H$ et de $T_i \cap H$. En sommant ces s inégalités, on obtient

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{s} \sum_{e \in \delta(T_i)} x_e \geq 3s.$$

Comme le terme de gauche est un entier pair, on obtient l'inégalité voulue. □

La solution fractionnaire x, représentée à la figure 21.11 (les arêtes e telles que $x_e = 0$ sont omises), est un exemple d'une inégalité de peigne violée dans K_{12} : considérons $H = \{1,2,3,4,5,6\}$, $T_1 = \{1,11\}$, $T_2 = \{2,12\}$ et $T_3 = \{5,6,7,8\}$. Il est facile de vérifier que l'inégalité de peigne correspondante est violée. Notons que les inégalités (21.1), (21.2), (21.3), (21.6) sont vérifiées et que x est optimum respectivement aux poids $c(e) := 1 - x_e$ (poids total 3), alors que le meilleur tour a un poids $\frac{7}{2}$.

Mentionnons juste une catégorie supplémentaire d'inégalités : les **inégalités d'arbre de cliques** :

Théorème 21.29. (Grötschel et Pulleyblank [1986]) *Soient $H_1, \ldots, H_r$ des sous-ensembles deux à deux disjoints de $V(G)$ (les poignées) et soient $T_1, \ldots, T_s$ ($s \geq 1$) des sous-ensembles propres non vides deux à deux disjoints de $V(G)$ (les dents) tels que :*

- *pour chaque poignée, le nombre de dents qu'elle intersecte est impair et supérieur ou égal à trois ;*
- *chaque dent T contient au moins un sommet n'appartenant pas à une poignée ;*
- *$G := K_n[H_1 \cup \cdots \cup H_r \cup T_1 \cup \cdots \cup T_s]$ est connexe, mais $G - (T_i \cap H_j)$ est non connexe dès que $T_i \cap H_j \neq \emptyset$.*

Notons t_j le nombre de poignées intersectant T_j ($j = 1, \ldots, s$). Alors tout $x \in Q(n)$ vérifie

$$\sum_{i=1}^{r} \sum_{e \in E(K_n[H_i])} x_e + \sum_{j=1}^{s} \sum_{e \in E(K_n[T_j])} x_e \leq \sum_{i=1}^{r} |H_i| + \sum_{j=1}^{s} (|T_j| - t_j) - \frac{s+1}{2}. \tag{21.8}$$

Nous omettons la preuve qui est techniquement assez difficile. Les inégalités d'arbre de cliques incluent (21.3) et (21.6) (exercice 14). Elles ont été généralisées, par exemple aux inégalités de bipartition par Boyd et Cunningham [1991]. Il existe un algorithme polynomial pour le PROBLÈME DE SÉPARATION des inégalités d'arbre de cliques (21.8) avec un nombre fixe de poignées et de dents (Carr [1997]), mais pas pour les inégalités d'arbre de cliques générales. Même pour le PROBLÈME DE SÉPARATION des inégalités de peigne, on ne connaît pas d'algorithme polynomial.

Les inégalités (21.1), (21.4) (restreintes au cas où $3 \leq |X| \leq n-3$) et (21.6) (restreintes au cas où F est un couplage) définissent des facettes distinctes de $Q(n)$ (pour $n \geq 6$). On prouve que les inégalités triviales (21.1) définissent des facettes en recherchant $\dim(Q(n))$ tours linéairement indépendants tels que $x_e = 1$ (et de même pour $x_e = 0$) pour une arête e fixée. La preuve est semblable à celle du théorème 21.23, nous renvoyons à Grötschel et Padberg [1979]. De même, toutes les inégalités (21.8) définissent des facettes de $Q(n)$ ($n \geq 6$). La preuve est assez compliquée, voir Grötschel et Padberg [1979] ou Grötschel et Pulleyblank [1986].

Le nombre de facettes de $Q(n)$ augmente rapidement : $Q(10)$ a déjà plus de 50 milliards de facettes. On ne connaît pas de description complète de $Q(n)$ et il paraît très improbable qu'il en existe une. Considérons le problème suivant :

FACETTES DU PVC	
Instance	Un entier n et une inégalité entière $ax \leq \beta$.
Question	L'inégalité considérée définit-elle une facette de $Q(n)$?

Le résultat suivant montre que l'existence d'une description complète du polytope du voyageur de commerce est peu probable :

Théorème 21.30. (Karp et Papadimitriou [1982]) *Si le problème des* FACETTES DU PVC *est dans NP, alors NP = coNP.*

De plus, décider si deux sommets donnés de $Q(n)$ sont adjacents (c.-à-d. appartiennent à une même face de dimension un) est un problème *NP*-complet (Papadimitriou [1978]).

21.5 Bornes inférieures

Supposons que nous ayons trouvé un tour à l'aide d'une heuristique, par exemple avec l'ALGORITHME DE LIN-KERNIGHAN. Nous ne savons pas par avance si ce tour est optimum ou au moins s'il est proche de l'optimum. Existe-t-il un moyen de garantir que notre tour ne s'écarte pas de plus de x pour cent de l'optimum ? Autrement dit, existe-t-il une borne inférieure pour l'optimum ?

On peut trouver des bornes inférieures en considérant une relaxation linéaire d'une formulation comme programme en nombres entiers du PVC, par exemple en prenant les inégalités (21.1), (21.2), (21.3), (21.6). Cependant, ce PL n'est pas facile à résoudre (bien qu'il existe un algorithme polynomial par la MÉTHODE DES ELLIPSOÏDES). Une borne inférieure plus raisonnable est obtenue en prenant seulement (21.1), (21.2), (21.6), c.-à-d. en recherchant un 2-couplage simple parfait de poids minimum (voir exercice 1 du chapitre 12).

Cependant, la méthode la plus efficace connue est d'utiliser une relaxation lagrangienne (paragraphe 5.6). La relaxation lagrangienne a été appliquée au PVC pour la première fois par Held et Karp [1970,1971]. Leur méthode est fondée sur la notion suivante :

Définition 21.31. *Soit un graphe complet K_n ayant pour ensemble de sommets $V(K_n) = \{1, \ldots, n\}$; un* **1-arbre** *est un graphe constitué d'un arbre couvrant sur les sommets $\{2, \ldots, n\}$ et de deux arêtes incidentes au sommet* 1.

Les tours sont précisément les 1-arbres T tels que $|\delta_T(i)| = 2$ pour $i = 1, \ldots, n$. Nous connaissons bien les arbres couvrants et les 1-arbres ne sont pas très différents. Par exemple, nous avons :

Proposition 21.32. *L'enveloppe convexe des vecteurs d'incidence de tous les 1-arbres est l'ensemble des vecteurs* $x \in [0,1]^{E(K_n)}$ *tels que*

$$\sum_{e \in E(K_n)} x_e = n,\ \sum_{e \in \delta(1)} x_e = 2,\ \sum_{e \in E(K_n[X])} x_e \leq |X| - 1\ (\emptyset \neq X \subseteq \{2,\dots,n\}).$$

Preuve. Cela est une conséquence directe du théorème 6.12. □

Observons que toute fonction objectif linéaire peut être facilement optimisée sur l'ensemble des 1-arbres : il suffit de rechercher un arbre couvrant de poids minimum sur $\{2,\dots,n\}$ (paragraphe 6.1) et de lui ajouter les deux arêtes les moins coûteuses incidentes au sommet 1. La relaxation lagrangienne fournit alors la borne inférieure suivante :

Proposition 21.33. (Held et Karp [1970]) *Étant donné une instance* (K_n, c) *du* PVC *avec* $V(K_n) = \{1,\dots,n\}$ *et* $\lambda = (\lambda_2,\dots,\lambda_n) \in \mathbb{R}^{n-1}$. *Alors*

$$LR(K_n, c, \lambda) := \min\left\{ c(E(T)) + \sum_{i=2}^{n} (|\delta_T(i)| - 2)\,\lambda_i : T \textit{ est un 1-arbre} \right\}$$

est une borne inférieure de la longueur d'un tour optimum, qui peut être calculée dans le temps nécessaire pour résoudre un PROBLÈME DE L'ARBRE COUVRANT MINIMUM *sur* $n-1$ *sommets.*

Preuve. Un tour optimum T est un 1-arbre tel que $|\delta_T(i)| = 2$ pour tout i, ce qui prouve que $LR(K_n, c, \lambda)$ est une borne inférieure. Étant donné $\lambda = (\lambda_2,\dots,\lambda_n)$, nous choisissons arbitrairement λ_1 et remplaçons les poids c par $c'((i,j)) := c((i,j)) + \lambda_i + \lambda_j$ $(1 \leq i < j \leq n)$. Alors il nous reste seulement à rechercher un 1-arbre de poids minimum respectivement à c'. □

Notons que les multiplicateurs de Lagrange λ_i $(i = 2,\dots,n)$ ne sont pas restreints aux nombres non négatifs, car les contraintes supplémentaires $|\delta_T(i)| = 2$ sont des égalités. Les λ_i peuvent être déterminés par la méthode des sous-gradients (voir paragraphe 5.6). La valeur maximum possible

$$HK(K_n, c) := \max\{LR(K_n, c, \lambda) : \lambda \in \mathbb{R}^{n-1}\}$$

(le dual lagrangien) est appelée la **borne de Held-Karp** pour (K_n, c). Nous avons :

Théorème 21.34. (Held et Karp [1970]) *Pour toute instance* (K_n, c) *du* PVC *avec* $V(K_n) = \{1,\dots,n\}$ *:*

$$HK(K_n, c) = \min\left\{ cx : 0 \leq x_e \leq 1\ (e \in E(K_n)),\ \sum_{e \in \delta(v)} x_e = 2\ (v \in V(K_n)),\right.$$
$$\left. \sum_{e \in E(K_n[I])} x_e \leq |I| - 1 \textit{ pour tout } \emptyset \neq I \subseteq \{2,\dots,n\} \right\}.$$

Preuve. Cela est une conséquence directe du théorème 5.36 et de la proposition 21.32. □

Autrement dit, la borne de Held-Karp est égale à la valeur optimale du PL associé au polytope des sous-tours (voir proposition 21.25). Cela nous est utile pour estimer la qualité de la borne de Held-Karp pour le PVC MÉTRIQUE. Nous utilisons également de nouveau l'idée de l'ALGORITHME DE CHRISTOFIDES :

Théorème 21.35. (Wolsey [1980]) *Pour toute instance du* PVC MÉTRIQUE*, la borne de Held-Karp est supérieure ou égale au $\frac{2}{3}$ de la longueur d'un tour optimum.*

Preuve. Soit (K_n, c) une instance du PVC MÉTRIQUE et soit T un 1-arbre de poids minimum dans (K_n, c). Nous avons

$$c(E(T)) \;=\; LR(K_n, c, 0) \;\leq\; HK(K_n, c).$$

Soit $W \subseteq V(K_n)$ le sous-ensemble constitué des sommets de degré impair dans T. Puisque chaque vecteur x dans le polytope des sous-tours de (K_n, c) vérifie $\sum_{e\in\delta(X)} x_e \geq 2$ pour tout $\emptyset \neq X \subset V(K_n)$, le polyèdre

$$\left\{ x : x_e \geq 0 \text{ pour } e \in E(K_n),\ \sum_{e\in\delta(X)} x_e \geq 2 \text{ pour tout } X,\ |X \cap W| \text{ impair} \right\}$$

contient le polytope des sous-tours. Ainsi, d'après le théorème 21.34,

$$\min \left\{ cx : x_e \geq 0 \text{ pour } e \in E(K_n),\ \sum_{e\in\delta(X)} x_e \geq 1 \text{ pour tout } X,\ |X \cap W| \text{ impair} \right\}$$
$$\leq \frac{1}{2} HK(K_n, c).$$

Mais observons maintenant que, d'après le théorème 12.16, le terme de gauche correspond au poids minimum d'un W-joint J dans (K_n, c). Donc $c(E(T)) + c(J) \leq \frac{3}{2} HK(K_n, c)$. Comme le graphe $G := (V(K_n), E(T) \cup J)$ est connexe et eulérien, il s'agit d'une borne supérieure de la longueur d'un tour optimum (d'après le lemme 21.3). □

Une preuve différente est due à Shmoys et Williamson [1990]. On ne sait pas si cette borne est serrée. L'instance de la figure 21.9 à la page 579 (les arêtes non représentées sont de poids égal à 3) est un exemple où la borne de Held-Karp (9) est strictement inférieure à la longueur du tour optimum (qui est égale à 10). Il existe des instances du PVC MÉTRIQUE où $\frac{HK(K_n,c)}{\mathrm{OPT}(K_n,c)}$ est arbitrairement proche de $\frac{3}{4}$ (exercice 15). Cependant, elles peuvent être considérées comme des exceptions : en pratique, la borne de Held-Karp est habituellement bien meilleure ; voir par exemple Johnson, McGeoch et Rothberg [1996] ou Applegate *et al.* [2007].

21.6 Méthodes par séparation et évaluation

La méthode par séparation et évaluation est une technique pour énumérer complètement toutes les solutions possibles sans avoir à les considérer une par une. Pour de nombreux problèmes d'optimisation combinatoire *NP*-difficiles, c'est la meilleure procédure connue pour obtenir une solution optimale. Elle a été proposée par Land et Doig [1960] et appliquée au PVC en premier par Little *et al.* [1963].

Pour appliquer la méthode par SÉPARATION ET ÉVALUATION à un problème d'optimisation combinatoire (disons de minimisation), nous avons besoin de deux étapes :

- «séparation» : un sous-ensemble donné des solutions possibles (des tours pour le PVC) peut être partitionné en au moins deux sous-ensembles non vides ;
- «évaluation» : pour un sous-ensemble obtenu après des séparations successives, on peut calculer une borne inférieure du coût d'une solution de ce sous-ensemble.

La procédure générale est alors la suivante :

SÉPARATION ET ÉVALUATION

Input Une instance d'un problème.

Output Une solution optimale S^*.

① Poser $T := (\{\mathcal{S}\}, \emptyset)$ pour l'arbre de départ, où $\mathcal{S}$ est l'ensemble de toutes les solutions réalisables.
Marquer $\mathcal{S}$ actif.
Poser $U := \infty$ pour la borne supérieure (ou appliquer une heuristique pour obtenir une meilleure borne supérieure).

② Choisir un sommet actif X de l'arbre T (s'il n'y en a aucun, **stop**).
Marquer X non actif.
(«séparation») Trouver une partition $X = X_1 \dot\cup \ldots \dot\cup X_t$.

③ **For** chaque $i = 1, \ldots, t$ **do** :
(«évaluation») Trouver une borne inférieure L du coût d'une solution dans X_i.
If $|X_i| = 1$ (disons $X_i = \{S\}$) et $\text{cost}(S) < U$ **then** :
Poser $U := \text{cost}(S)$ et $S^* := S$.
If $|X_i| > 1$ et $L < U$ **then** :
Poser $T := (V(T) \cup \{X_i\}, E(T) \cup \{\{X, X_i\}\})$ et marquer X_i actif.

④ **Go to** ②.

Il devrait être clair que la méthode précédente trouve toujours une solution optimale. Bien entendu, l'implémentation (et l'efficacité) dépend beaucoup du problème considéré. Nous présenterons une implémentation possible pour le PVC.

La façon la plus simple d'effectuer la séparation est de choisir une arête e et de noter $X = X_e \cup Y_e$, où X_e (resp. Y_e) est constitué des solutions dans X qui contiennent (resp. ne contiennent pas) l'arête e. On peut alors noter tout sommet X de l'arbre sous la forme

$$\mathcal{S}_{A,B} = \{S \in \mathcal{S} : A \subseteq S, B \cap S = \emptyset\} \qquad \text{pour } A, B \subseteq E(G).$$

Pour ces sommets $X = \mathcal{S}_{A,B}$, le PVC, avec les contraintes supplémentaires que toutes les arêtes de A, mais aucune de B, appartiennent au tour, peut être considéré comme un PVC non contraint en modifiant les poids c en conséquence : pour cela nous posons

$$c'_e := \begin{cases} c_e & \text{si } e \in A \\ c_e + C & \text{si } e \notin A \cup B \\ c_e + 2C & \text{si } e \in B \end{cases}$$

avec $C := \sum_{e \in E(G)} c_e + 1$. Alors les tours dans $\mathcal{S}_{A,B}$ sont exactement les tours dont le poids modifié est inférieur à $(n + 1 - |A|)C$. De plus, le poids original et le poids modifié d'un tour dans $\mathcal{S}_{A,B}$ diffèrent d'exactement $(n - |A|)C$.

Alors la borne de Held-Karp (voir paragraphe 21.5) peut être utilisée pour implémenter l'étape d'«évaluation».

La méthode par SÉPARATION ET ÉVALUATION précédente pour le PVC a été utilisée pour résoudre des instances d'assez grande taille du PVC (jusqu'à environ cent villes).

L'algorithme de SÉPARATION ET ÉVALUATION est également souvent utilisé pour résoudre des programmes en nombres entiers, en particulier lorsque les variables sont binaires (restreintes à 0 ou 1). Dans ce cas, l'étape de séparation la plus naturelle consiste à prendre une variable et à essayer les deux valeurs possibles pour elle. Une borne inférieure peut être facilement déterminée en résolvant la relaxation linéaire correspondante.

Dans le pire des cas, l'algorithme de SÉPARATION ET ÉVALUATION n'est pas meilleur que l'énumération complète de toutes les solutions possibles. En pratique, l'efficacité ne dépend pas que de la manière dont les étapes de «séparation»et d'«évaluation»sont implémentées. Il est également important d'avoir une bonne stratégie pour choisir le sommet actif X à l'étape ② de l'algorithme. De plus, une bonne heuristique au départ (et ainsi une bonne borne supérieure pour commencer) peut aider à ce que l'arbre de séparation et évaluation T reste petit.

L'algorithme de SÉPARATION ET ÉVALUATION est souvent combiné avec une méthode des plans coupants (voir paragraphe 5.5), fondée sur les résultats du paragraphe 21.4. On procède de la manière suivante. Puisque l'on a un nombre exponentiel de contraintes (qui ne décrivent même pas $Q(n)$ complètement), on commence par résoudre le PL

$$\min \left\{ cx : 0 \leq x_e \leq 1 \ (e \in E(K_n)), \sum_{e \in \delta(v)} x_e = 2 \ (v \in V(K_n)) \right\},$$

c.-à-d. avec les contraintes (21.1) et (21.2). Le polyèdre correspondant contient les vecteurs entiers associés aux 2-couplages simples parfaits. Supposons que nous ayons une solution x^* du PL précédent. Il y a trois cas :

(a) x^* est le vecteur d'incidence d'un tour.

(b) On trouve une inégalité de sous-tour (21.3), une inégalité de 2-couplage (21.6), une inégalité de peigne (21.7) ou une inégalité d'arbre de cliques (21.8) violée.

(c) On ne peut trouver d'inégalité violée (en particulier aucune inégalité de sous-tour n'est violée), mais x^* n'est pas entier.

Si x^* est entier, mais n'est pas le vecteur d'incidence d'un tour, une inégalité de sous-tour doit être violée d'après la proposition 21.24.

Dans le cas (a), nous avons terminé. Dans le cas (b), nous ajoutons simplement l'inégalité violée (ou éventuellement plusieurs inégalités violées) à notre PL et nous résolvons le nouveau PL. Dans le cas (c) nous avons une borne inférieure (en général très bonne) de la longueur d'un tour. En utilisant cette borne (et la solution fractionnaire), nous pouvons démarrer une procédure de SÉPARATION ET ÉVALUATION. Grâce à la borne inférieure, nous pouvons, avec un peu de chance, fixer à l'avance de nombreuses variables et ainsi réduire considérablement les étapes de séparation nécessaires pour obtenir une solution optimale. De plus, à chaque nœud de l'arbre de séparation et évaluation, nous pouvons de nouveau rechercher des inégalités violées.

Cette méthode de **séparation et coupe** (en anglais *branch and cut*) a été utilisée pour résoudre des instances du PVC avec plus de 10 000 villes jusqu'à l'optimalité. Bien entendu, de nombreuses idées sophistiquées, que nous n'avons pas décrites ici, sont nécessaires pour obtenir une implémentation efficace. En particulier, de bonnes heuristiques sont essentielles pour détecter les inégalités violées. Voir Applegate *et al.* [2003, 2007] et Jünger et Naddef [2001] pour davantage d'informations et d'autres références.

Ces succès, dans la résolution d'instances de grande taille, contrastent avec des temps de calcul (dans le pire des cas) décevants. Woeginger [2002] présente une étude d'algorithmes exacts sous-exponentiels pour des problèmes *NP*-difficiles ; voir aussi l'exercice 1.

Exercices

1. Décrire un algorithme exact pour le PVC à l'aide de la programmation dynamique. Si les sommets (villes) sont numérotés $1, \ldots, n$, nous notons $\gamma(A, x)$ le coût minimum d'une chaîne P de 1 à x telle que $V(P) = A \cup \{1\}$, pour tout $A \subseteq \{2, 3, \ldots, n\}$ et $x \in A$. L'idée est de calculer maintenant tous les nombres $\gamma(A, x)$. Comparer le temps de calcul de cet algorithme avec l'énumération simple de tous les tours.
   (Held et Karp [1962])
   Remarque : cela est l'algorithme exact pour le PVC ayant le meilleur temps de calcul (dans le pire des cas) connu. Pour le PVC EUCLIDIEN, Hwang, Chang

et Lee [1993] ont décrit un algorithme exact, utilisant des séparateurs planaires, avec un temps de calcul sous-exponentiel $O(c^{\sqrt{n}\log n})$.

2. Supposons que les n villes d'une instance du PVC soient partitionnées en m groupes de telle façon que la distance entre deux villes soit nulle si et seulement si elles appartiennent au même groupe.
   (a) Prouver qu'il existe un tour optimum avec au plus $m(m-1)$ arêtes de poids positif.
   (b) Prouver qu'un tel PVC peut être résolu en temps polynomial si m est fixé.

   (Triesch, Nolles et Vygen [1994])

3. Considérons le problème suivant. Un camion partant d'un dépôt d_1 doit passer chez des clients $c_1, \ldots, c_n$ et finalement revenir à d_1. Entre les passages chez deux clients, il doit se rendre dans l'un des dépôts $d_1, \ldots, d_k$. Étant donné des distances symétriques non négatives entre les clients et les dépôts, nous recherchons le tour le plus court possible.
   (a) Montrer que ce problème est *NP*-complet.
   (b) Montrer qu'il peut être résolu en temps polynomial si k est fixé. (*Indication :* utiliser l'exercice 2.)

   (Triesch, Nolles et Vygen [1994])

4. Considérons le PVC ASYMÉTRIQUE vérifiant l'inégalité triangulaire : étant donné un nombre $n \in \mathbb{N}$ et des distances $c((i,j)) \in \mathbb{R}_+$ pour $i, j \in \{1, \ldots, n\}$, $i \neq j$, vérifiant l'inégalité triangulaire $c((i,j)) + c((j,k)) \geq c((i,k))$ pour tout $i, j, k \in \{1, \ldots, n\}$, trouver une permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ telle que $\sum_{i=1}^{n-1} c((\pi(i), \pi(i+1))) + c((\pi(n), \pi(1)))$ soit minimum.
   Décrire un algorithme qui trouve toujours une solution dont le coût est égal à au plus $\log n$ fois l'optimum.
   Indication : trouver d'abord un graphe orienté H tel que $V(H) = \{1, \ldots, n\}$, $|\delta_H^-(v)| = |\delta_H^+(v)| = 1$ pour tout $v \in V(H)$ et de coût minimum $c(E(H))$. Contracter les circuits de H et itérer.
   (Frieze, Galbiati et Maffioli [1982])
   Remarque : le meilleur algorithme actuel, dû à Kaplan *et al.* [2005], est un algorithme d'approximation de facteur $(0,842 \log n)$.

* 5. Trouver des instances du PVC EUCLIDIEN pour lesquelles l'ALGORITHME ARBRE-DOUBLE retourne un tour dont la longueur est arbitrairement proche de deux fois l'optimum.

6. Soit G un graphe biparti complet de bipartition $V(G) = A \dot\cup B$, où $|A| = |B|$. Soit $c : E(G) \to \mathbb{R}_+$ une fonction coût telle que $c((a,b)) + c((b,a')) + c((a',b')) \geq c((a,b'))$ pour tout $a, a' \in A$ et $b, b' \in B$. L'objectif est de trouver un cycle hamiltonien dans G de coût minimum. Ce problème est appelé le PVC BIPARTI MÉTRIQUE.
   (a) Prouver que, pour tout k, s'il existe un algorithme d'approximation de facteur k pour le PVC BIPARTI MÉTRIQUE, alors il existe aussi un algorithme d'approximation de facteur k pour le PVC MÉTRIQUE.

(b) Trouver un algorithme d'approximation de facteur 2 pour le PVC BIPARTI MÉTRIQUE. (*Indication :* combiner l'exercice 25 du chapitre 13 avec l'idée de l'ALGORITHME ARBRE-DOUBLE.)

(Frank *et al.* [1998], Chalasani, Motwani et Rao [1996])

∗ 7. Trouver des instances du PVC MÉTRIQUE pour lesquelles l'ALGORITHME DE CHRISTOFIDES retourne un tour dont la longueur est arbitrairement proche de $\frac{3}{2}$ fois l'optimum.

8. Montrer que les résultats du paragraphe 21.2 s'étendent au PROBLÈME DE L'ARBRE DE STEINER EUCLIDIEN. Décrire un schéma d'approximation pour ce problème.

9. Prouver que, dans l'ALGORITHME DE LIN-KERNIGHAN, un ensemble X_i ne contient jamais plus d'un élément pour tout nombre impair i tel que $i > p_2 + 1$.

10. Considérons le problème de décision suivant :

UN AUTRE CYCLE HAMILTONIEN	
Instance	Un graphe G et un cycle hamiltonien C dans G.
Tâche:	Existe-t-il un autre cycle hamiltonien dans G ?

(a) Montrer que ce problème est *NP*-complet. (*Indication :* revoir la preuve du lemme 21.18.)

∗ (b) Prouver que pour un graphe 3-régulier G et $e \in E(G)$, le nombre de cycles hamiltoniens contenant e est pair.

(c) Montrer que le problème d'UN AUTRE CYCLE HAMILTONIEN pour les graphes 3-réguliers est dans *P*. (Néanmoins, étant donné un graphe 3-régulier G et un cycle hamiltonien C dans G, on ne connaît pas d'algorithme polynomial pour trouver un autre cycle hamiltonien.)

11. Soit $(X, (S_x)_{x \in X}, x, \mathrm{but})$ un problème d'optimisation discrète avec des voisinages $N_x(y) \subseteq S_x$ pour $y \in S_x$ et $x \in X$. Supposons que nous puissions trouver, pour chaque $x \in X$, un élément de S_x en temps polynomial et que nous puissions trouver, pour chaque $y \in S_x$, un élément $y' \in N_x(y)$ de meilleur coût ou conclure qu'il n'en existe pas. Alors, on dit que le problème avec ces voisinages appartient à la classe *PLS* (recherche locale polynomiale, en anglais *polynomial local search*). Prouver que, s'il existe un problème dans *PLS* pour lequel il est *NP*-difficile de calculer un optimum local pour une instance donnée, alors *NP* = *coNP*.
Indication : concevoir un algorithme non déterministe pour tout problème *coNP*-complet.
(Johnson, Papadimitriou et Yannakakis [1988])
Remarque : le PVC avec le voisinage k-opt et avec le voisinage de Lin-Kernighan est *PLS*-complet (Krentel [1989], Papadimitriou [1992]), c.-à-d. que si l'on peut trouver un optimum en temps polynomial, on peut le faire également pour tout problème et tout voisinage dans *PLS* (et cela impliquerait une autre preuve du théorème 4.18 due à l'exactitude de l'ALGORITHME DU SIMPLEXE).

12. Montrer que l'on peut optimiser toute fonction linéaire sur le polytope défini par (21.1), (21.2), (21.3), (21.6).
Indication : utiliser le théorème 21.23 pour réduire la dimension afin d'obtenir un polytope de pleine dimension. Trouver un point à l'intérieur et appliquer le théorème 4.21.

13. Considérons les inégalités de 2-couplage (21.6) de la proposition 21.27. Montrer qu'il est suffisant de considérer les inégalités (21.6) dans le cas où F est un couplage, les autres inégalités de 2-couplage étant impliquées par celles-ci.

14. Montrer que les inégalités de sous-tours (21.3), les inégalités de 2-couplage (21.6) et les inégalités de peigne (21.7) sont des cas particuliers des inégalités d'arbre de cliques (21.8).

15. Prouver qu'il existe des instances (K_n, c) du PVC MÉTRIQUE telles que $\frac{HK(K_n,c)}{\mathrm{OPT}(K_n,c)}$ soit arbitrairement proche de $\frac{3}{4}$.
Indication : remplacer les arêtes de poids 1 de la figure 21.9 par de longues chaînes et considérer la fermeture métrique.

16. Considérons le PVC avec n villes. Pour toute fonction poids $w : E(K_n) \to \mathbb{R}_+$, notons c^*_w la longueur d'un tour optimum respectivement à w. Prouver : si $L_1 \leq c^*_{w_1}$ et $L_2 \leq c^*_{w_2}$ pour deux fonctions poids w_1 et w_2, alors $L_1 + L_2 \leq c^*_{w_1+w_2}$, où la somme des deux fonctions poids est prise composante par composante.

17. Soit c_0 le coût du tour optimum pour une instance avec n villes du PVC MÉTRIQUE et soit c_1 le coût du deuxième meilleur tour. Montrer que

$$\frac{c_1 - c_0}{c_0} \;\leq\; \frac{2}{n}.$$

(Papadimitriou et Steiglitz [1978])

18. Soit $x \in [0,1]^{E(K_n)}$ tel que $\sum_{e\in\delta(v)} x_e = 2$ pour tout $v \in V(K_n)$. Prouver que s'il existe une contrainte de sous-tour violée, c.-à-d. un ensemble $S \subset V(K_n)$ tel que $\sum_{e\in\delta(S)} x_e < 2$, alors il en existe un tel que $x_e < 1$ pour tout $e \in \delta(S)$. (Crowder et Padberg [1980])

19. Soient une famille $\mathcal{F}$ de sous-ensembles (non nécessairement distincts) de $\{1,\ldots,n\}$ et un vecteur $x \in \mathbb{R}^{E(K_n)}$, définissons $\mathcal{F}(x) := \sum_{X\in\mathcal{F}} \sum_{e\in\delta(X)} x_e$ et notons $\mu_{\mathcal{F}}$ le minimum de $\mathcal{F}(x)$ pris sur tous les vecteurs d'incidence des tours dans K_n. Une inégalité de la forme $\mathcal{F}(x) \geq \mu_{\mathcal{F}}$ est appelée une *inégalité d'hypergraphe*. (21.5) et (21.7) en sont des exemples.
Montrer que le polytope du PVC peut être décrit par des contraintes de degré et des inégalités d'hypergraphe, c.-à-d. montrer qu'il existe des familles $\mathcal{F}_1,\ldots,\mathcal{F}_k$ telles que $Q(n) =$

$$\left\{ x \in \mathbb{R}^{E(K_n)} : \sum_{e\in\delta(v)} x_e = 2 \,(v \in V(K_n)),\ \mathcal{F}_i(x) \leq \mu_{\mathcal{F}_i} \,(i = 1,\ldots,k) \right\}.$$

Indication : réécrire toute inégalité induisant une facette en utilisant le fait que $\sum_{e\in\delta(\{v,w\})} x_e = 4 - 2x_{(v,w)}$ pour chaque x vérifiant les contraintes de degré. (Applegate *et al.* [2007])

Références

Littérature générale :

Applegate, D.L., Bixby, R., Chvátal, V., Cook, W.J. [2007] : The Traveling Salesman Problem : A Computational Study. Princeton University Press 2007

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. [1998] : Combinatorial Optimization. Wiley, New York 1998, Chapter 7

Gutin, G., Punnen, A.P. [2002] : The Traveling Salesman Problem and Its Variations. Kluwer, Dordrecht 2002

Jungnickel, D. [1999] : Graphs, Networks and Algorithms. Springer, Berlin 1999, Chapter 14

Lawler, E.L., Lenstra J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. [1985] : The Traveling Salesman Problem. Wiley, Chichester 1985

Jünger, M., Reinelt, G., Rinaldi, G. [1995] : The traveling salesman problem. In : Handbooks in Operations Research and Management Science ; Volume 7 ; Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995

Papadimitriou, C.H., Steiglitz, K. [1982] : Combinatorial Optimization ; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Section 17.2, Chapters 18 and 19

Reinelt, G. [1994] : The Traveling Salesman ; Computational Solutions for TSP Applications. Springer, Berlin 1994

Références citées :

Aarts, E., Lenstra, J.K. [1997] : Local Search in Combinatorial Optimization. Wiley, Chichester 1997

Applegate, D., Bixby, R., Chvátal, V., Cook, W. [2003] : Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. Mathematical Programming B 97 (2003), 91–153

Applegate, D., Cook, W., Rohe, A. [2003] : Chained Lin-Kernighan for large traveling salesman problems. INFORMS Journal on Computing 15 (2003), 82–92

Arora, S. [1998] : Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. Journal of the ACM 45 (1998), 753–782

Berman, P., Karpinski, M. [2006] : 8/7-approximation algorithm for (1,2)-TSP. Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (2006), 641–648

Boyd, S.C., Cunningham, W.H. [1991] : Small traveling salesman polytopes. Mathematics of Operations Research 16 (1991), 259–271

Burkard, R.E., Deĭneko, V.G., Woeginger, G.J. [1998] : The travelling salesman and the PQ-tree. Mathematics of Operations Research 23 (1998), 613–623

Carr, R. [1997] : Separating clique trees and bipartition inequalities having a fixed number of handles and teeth in polynomial time. Mathematics of Operations Research 22 (1997), 257–265

Chalasani, P., Motwani, R., Rao, A. [1996] : Algorithms for robot grasp and delivery. Proceedings of the 2nd International Workshop on Algorithmic Foundations of Robotics (1996), 347–362

Chandra, B., Karloff, H., Tovey, C. [1999] : New results on the old k-opt algorithm for the traveling salesman problem. SIAM Journal on Computing 28 (1999), 1998–2029

Christofides, N. [1976] : Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh 1976

Chvátal, V. [1973] : Edmonds' polytopes and weakly hamiltonian graphs. Mathematical Programming 5 (1973), 29–40

Crowder, H., Padberg, M.W. [1980] : Solving large-scale symmetric travelling salesman problems to optimality. Management Science 26 (1980), 495–509

Dantzig, G., Fulkerson, R., Johnson, S. [1954] : Solution of a large-scale traveling-salesman problem. Operations Research 2 (1954), 393–410

Englert, M., Röglin, H., Vöcking, B. [2007] : Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (2007), 1295–1304

Frank, A., Triesch, E., Korte, B., Vygen, J. [1998] : On the bipartite travelling salesman problem. Report No. 98866, Research Institute for Discrete Mathematics, University of Bonn, 1998

Frieze, A., Galbiati, G., Maffioli, F. [1982] : On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks 12 (1982), 23–39

Garey, M.R., Graham, R.L., Johnson, D.S. [1976] : Some NP-complete geometric problems. Proceedings of the 8th Annual ACM Symposium on the Theory of Computing (1976), 10–22

Grötschel, M., Padberg, M.W. [1979] : On the symmetric travelling salesman problem. Mathematical Programming 16 (1979), 265–302

Grötschel, M., Pulleyblank, W.R. [1986] : Clique tree inequalities and the symmetric travelling salesman problem. Mathematics of Operations Research 11 (1986), 537–569

Held, M., Karp, R.M. [1962] : A dynamic programming approach to sequencing problems. Journal of SIAM 10 (1962), 196–210

Held M., Karp, R.M. [1970] : The traveling-salesman problem and minimum spanning trees. Operations Research 18 (1970), 1138–1162

Held, M., Karp, R.M. [1971] : The traveling-salesman problem and minimum spanning trees ; part II. Mathematical Programming 1 (1971), 6–25

Hurkens, C.A.J., Woeginger, G.J. [2004] : On the nearest neighbour rule for the traveling salesman problem. Operations Research Letters 32 (2004), 1–4

Hwang, R.Z., Chang, R.C., Lee, R.C.T. [1993] : The searching over separators strategy to solve some *NP*-hard problems in subexponential time. Algorithmica 9 (1993), 398–423

Johnson, D.S., McGeoch, L.A., Rothberg, E.E. [1996] : Asymptotic experimental analysis for the Held-Karp traveling salesman bound. Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (1996), 341–350

Johnson, D.S., Papadimitriou, C.H., Yannakakis, M. [1988] : How easy is local search ? Journal of Computer and System Sciences 37 (1988), 79–100

Jünger, M. Naddef, D. [2001] : Computational Combinatorial Optimization. Springer, Berlin 2001

Kaplan, H., Lewenstein, M., Shafrir, N., Sviridenko, M. [2005] : Approximation algorithms for the asymmetric TSP by decomposing directed regular multigraphs. Journal of the ACM 52 (2005), 602–626

Karp, R.M. [1977] : Probabilistic analysis of partitioning algorithms for the TSP in the plane. Mathematics of Operations Research 2 (1977), 209–224

Karp, R.M., Papadimitriou, C.H. [1982] : On linear characterizations of combinatorial optimization problems. SIAM Journal on Computing 11 (1982), 620–632

Klein, P.N. [2005] : A linear-time approximation scheme for planar weighted TSP. Proceedings of the 46th Annual IEEE Symposium on the Foundations of Computer Science (2005), 647–657

Krentel, M.W. [1989] : Structure in locally optimal solutions. Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (1989), 216–221

Land, A.H., Doig, A.G. [1960] : An automatic method of solving discrete programming problems. Econometrica 28 (1960), 497–520

Lin, S., Kernighan, B.W. [1973] : An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21 (1973), 498–516

Little, J.D.C., Murty, K.G., Sweeny, D.W., Karel, C. [1963] : An algorithm for the traveling salesman problem. Operations Research 11 (1963), 972–989

Michiels, W., Aarts, E., Korst, J. [2007] : Theoretical Aspects of Local Search. Springer, Berlin 2007

Mitchell, J. [1999] : Guillotine subdivisions approximate polygonal subdivisions : a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. SIAM Journal on Computing 28 (1999), 1298–1309

Papadimitriou, C.H. [1977] : The Euclidean traveling salesman problem is *NP*-complete. Theoretical Computer Science 4 (1977), 237–244

Papadimitriou, C.H. [1978] : The adjacency relation on the travelling salesman polytope is *NP*-complete. Mathematical Programming 14 (1978), 312–324

Papadimitriou, C.H. [1992] : The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. SIAM Journal on Computing 21 (1992), 450–465

Papadimitriou, C.H., Steiglitz, K. [1977] : On the complexity of local search for the traveling salesman problem. SIAM Journal on Computing 6 (1), 1977, 76–83

Papadimitriou, C.H., Steiglitz, K. [1978] : Some examples of difficult traveling salesman problems. Operations Research 26 (1978), 434–443

Papadimitriou, C.H., Vempala, S. [2006] : On the approximability of the traveling salesman problem. Combinatorica 26 (2006), 101–120

Papadimitriou, C.H., Yannakakis, M. [1993] : The traveling salesman problem with distances one and two. Mathematics of Operations Research 18 (1993), 1–12

Rao, S.B., Smith, W.D. [1998] : Approximating geometric graphs via “spanners” and “banyans”. Proceedings of the 30th Annual ACM Symposium on Theory of Computing (1998), 540–550

Rosenkrantz, D.J. Stearns, R.E., Lewis, P.M. [1977] : An analysis of several heuristics for the traveling salesman problem. SIAM Journal on Computing 6 (1977), 563–581

Sahni, S., Gonzalez, T. [1976] : *P*-complete approximation problems. Journal of the ACM 23 (1976), 555–565

Shmoys, D.B., Williamson, D.P. [1990] : Analyzing the Held-Karp TSP bound : a monotonicity property with application. Information Processing Letters 35 (1990), 281–285

Triesch, E., Nolles, W., Vygen, J. [1994] : Die Einsatzplanung von Zementmischern und ein Traveling Salesman Problem In : Operations Research ; Reflexionen aus Theorie und Praxis (B. Werners, R. Gabriel, eds.), Springer, Berlin 1994 [in German]

Woeginger, G.J. [2002] : Exact algorithms for *NP*-hard problems. OPTIMA 68 (2002), 2–8

Wolsey, L.A. [1980] : Heuristic analysis, linear programming and branch and bound. Mathematical Programming Study 13 (1980), 121–134

Chapitre 22

Le problème de localisation

De nombreux problèmes de décisions économiques requièrent de sélectionner et/ou de placer certaines installations afin de répondre efficacement à des demandes données. On peut par exemple s'intéresser au placement d'usines, d'installations de stockage, de dépôts, d'entrepôts, de bibliothèques, de casernes de pompiers, d'hôpitaux, d'antennes pour les services sans fil (comme la radiodiffusion télévisuelle ou les services de téléphonie mobile), etc. Ces problèmes ont en commun le fait que l'on doit choisir un ensemble d'installations, chacune ayant une certaine position, avec pour objectif de satisfaire au mieux les demandes (des clients, des utilisateurs, etc). Les problèmes de localisation, qui apparaissent également dans des contextes moins évidents, ont de nombreuses applications.

Le modèle le plus étudié en localisation discrète est le PROBLÈME DE LOCALISATION SANS CAPACITÉS (en anglais *uncapacitated facility location problem*), également connu sous le nom de problème de localisation d'usines ou problème de localisation d'entrepôts. Il est présenté au paragraphe 22.1. Bien qu'il ait été étudié intensivement depuis les années soixante (voir, par exemple, Stollsteimer [1963], Balinski et Wolfe [1963], Kuehn et Hamburger [1963], Manne [1964]), on n'en connaissait pas d'algorithme d'approximation jusqu'en 1997. Depuis lors, plusieurs approches assez différentes ont été utilisées pour trouver une borne supérieure du rapport d'approximation. Nous les présenterons dans ce chapitre et nous considérerons également des extensions à des problèmes plus généraux, comme le problème de localisation avec capacités, le PROBLÈME DU k-MÉDIAN et le PROBLÈME DE LOCALISATION UNIVERSEL.

22.1 Problème de localisation sans capacités

Le problème de base pour lequel nous allons présenter de nombreux résultats est le PROBLÈME DE LOCALISATION SANS CAPACITÉS. Il est défini de la façon suivante :

PROBLÈME DE LOCALISATION SANS CAPACITÉS

Instance Un ensemble fini $\mathcal{D}$ de clients, un ensemble fini $\mathcal{F}$ d'installations (potentielles), un coût fixé $f_i \in \mathbb{R}_+$ pour construire l'installation $i \in \mathcal{F}$ et un coût de fonctionnement $c_{ij} \in \mathbb{R}_+$ pour chaque $i \in \mathcal{F}$ et $j \in \mathcal{D}$.

Tâche Trouver un sous-ensemble X d'installations (dites **ouvertes**) et une affectation $\sigma : \mathcal{D} \to X$ des clients aux installations ouvertes, telle que la somme des coûts de construction et des coûts de fonctionnement

$$\sum_{i \in X} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j}$$

soit minimum.

Dans de nombreuses applications pratiques, les coûts de fonctionnement correspondent à une métrique c sur $\mathcal{D} \cup \mathcal{F}$ (par exemple, lorsqu'ils sont proportionnels aux distances ou au temps de parcours). Dans ce cas, nous avons

$$c_{ij} + c_{i'j} + c_{i'j'} \geq c_{ij'} \quad \text{pour tout } i, i' \in \mathcal{F} \text{ et } j, j' \in \mathcal{D}. \tag{22.1}$$

Inversement, si cette condition est vérifiée, nous pouvons définir $c_{ii} := 0$ et $c_{ii'} := \min_{j \in \mathcal{D}}(c_{ij} + c_{i'j})$ pour $i, i' \in \mathcal{F}$, $c_{jj} := 0$ et $c_{jj'} := \min_{i \in \mathcal{F}}(c_{ij} + c_{ij'})$ pour $j, j' \in \mathcal{D}$ et $c_{ji} := c_{ij}$ pour $j \in \mathcal{D}$ et $i \in \mathcal{F}$ et obtenir une (semi-)métrique c sur $\mathcal{D} \cup \mathcal{F}$. Ainsi, nous parlons de coûts de fonctionnement *métriques* si (22.1) est vérifié. Le problème précédent, restreint aux instances avec des coûts de fonctionnement métriques, est appelé le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS.

Proposition 22.1. *Le* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS *est fortement NP-difficile.*

Preuve. Nous considérons le PROBLÈME DE COUVERTURE D'ENSEMBLES DE POIDS MINIMUM avec des poids unitaires (qui est fortement *NP*-difficile d'après le corollaire 15.24). Toute instance $(U, \mathcal{S})$ peut être transformée en une instance du PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS de la façon suivante : considérons $\mathcal{D} := U$, $\mathcal{F} := \mathcal{S}$, $f_i := 1$ pour $i \in \mathcal{S}$, $c_{ij} := 1$ pour $j \in i \in \mathcal{S}$ et $c_{ij} := 3$ pour $j \in U \setminus \{i\}$, $i \in \mathcal{S}$. Alors, pour $k \leq |\mathcal{S}|$, l'instance résultante a une solution de coût $|\mathcal{D}| + k$ si et seulement si $(U, \mathcal{S})$ a une couverture d'ensembles de cardinalité k. □

Dans la preuve précédente, le nombre 3 peut être remplacé par tout nombre supérieur à 1 mais pas supérieur à 3 (sinon la proposition 22.1 ne serait pas satisfaite). En fait, une construction similaire montre que des coûts de fonctionnement métriques sont nécessaires pour obtenir des algorithmes d'approximation : si dans la preuve précédente, nous posons $c_{ij} := \infty$ pour $j \in U \setminus \{i\}$ et $i \in \mathcal{S}$ nous voyons que tout algorithme d'approximation pour le PROBLÈME DE LOCALISATION SANS CAPACITÉS impliquerait un algorithme d'approximation pour la couverture

d'ensembles avec le même rapport de performance (et il n'existe pas d'algorithme d'approximation de facteur constant pour la couverture d'ensembles, à moins que $P = NP$; voir paragraphe 16.1). Guha et Khuller [1999] et Sviridenko [non publié] ont étendu la construction précédente pour montrer qu'un algorithme d'approximation de facteur $1,463$ pour le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS (même avec des coûts de fonctionnement égaux à 1 et 3 seulement) impliquerait que $P = NP$ (voir Vygen [2005] pour plus de détails).

Inversement, considérons une instance donnée du PROBLÈME DE LOCALISATION SANS CAPACITÉS. En posant $U := \mathcal{D}$, $\mathcal{S} = 2^{\mathcal{D}}$ et $c(D) := \min_{i\in\mathcal{F}}(f_i + \sum_{j\in D} c_{ij})$ pour $D \subseteq \mathcal{D}$, on obtient une instance équivalente du PROBLÈME DE COUVERTURE D'ENSEMBLES DE POIDS MINIMUM. Bien que cette instance soit de taille exponentielle, nous pouvons exécuter l'ALGORITHME GLOUTON DE COUVERTURE D'ENSEMBLES et obtenir une solution de coût au plus égal à $(1+\frac{1}{2}+\cdots+\frac{1}{|\mathcal{D}|})$ fois l'optimum en temps polynomial (voir théorème 16.3), comme proposé par Hochbaum [1982] :

À chaque étape, nous devons trouver une paire $(D, i) \in 2^{\mathcal{D}} \times \mathcal{F}$ telle que $\frac{f_i+\sum_{j\in D} c_{ij}}{|D|}$ soit minimum, ouvrir l'usine i, affecter tous les clients dans D à i et ne plus les considérer ensuite. Bien qu'il y ait un nombre exponentiel de choix, il est facile d'en trouver un meilleur puisqu'il suffit de considérer les paires (D^i_k, i) pour $i \in \mathcal{F}$ et $k \in \{1, \ldots, |\mathcal{D}|\}$, où D^i_k est l'ensemble des k premiers clients pris par ordre croissant respectivement à c_{ij}. De manière évidente, les autres paires ne peuvent pas être meilleures.

Jain *et al.* [2003] ont montré que la garantie de performance de cet algorithme glouton est $\Omega(\log n/\log\log n)$ même pour des instances métriques, où $n = |\mathcal{D}|$. En fait, avant l'article de Shmoys, Tardos et Aardal [1997], on ne connaissait pas d'algorithme d'approximation de facteur constant, même avec des coûts de fonctionnement métriques. Depuis lors, la situation a nettement changé. Dans les paragraphes suivants, nous présentons différentes techniques permettant d'obtenir des approximations de facteur constant pour le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS.

Un problème encore plus restreint apparaît dans le cas particulier où les installations et les clients sont des points dans le plan et où les coûts de fonctionnement correspondent aux distances. Dans ce cas, Arora, Raghavan et Rao [1998] ont montré que le problème a un schéma d'approximation, c.-à-d. un algorithme d'approximation de facteur k pour tout $k > 1$, similaire à l'algorithme du paragraphe 21.2. Ce résultat a été amélioré par Kolliopoulos et Rao [2007], mais l'algorithme semble être encore trop lent en pratique.

Dans le reste de ce chapitre, nous supposons que nous avons des coûts de fonctionnement métriques. Pour une instance donnée $\mathcal{D}, \mathcal{F}, f_i, c_{ij}$ et un sous-ensemble non vide donné X d'installations, on peut calculer facilement une meilleure affectation possible $\sigma : \mathcal{D} \to X$ vérifiant $c_{\sigma(j)j} = \min_{i\in X} c_{ij}$. Ainsi, nous appellerons souvent un ensemble non vide $X \subseteq \mathcal{F}$ une solution réalisable, avec coût d'installation $c_F(X) := \sum_{i\in X} f_i$ et coût de fonctionnement $c_S(X) := \sum_{j\in\mathcal{D}} \min_{i\in X} c_{ij}$.

L'objectif est de trouver un sous-ensemble non vide $X \subseteq \mathcal{F}$ tel que $c_F(X)+c_S(X)$ soit minimum.

22.2 Solutions arrondies de la programmation linéaire

Le PROBLÈME DE LOCALISATION SANS CAPACITÉS peut être formulé comme un PL en nombre entiers de la manière suivante :

$$\begin{array}{lrcll}
\min & \displaystyle\sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} & & & \\
\text{s.c.} & x_{ij} & \leq & y_i & (i\in\mathcal{F}, j\in\mathcal{D}) \\
& \displaystyle\sum_{i\in\mathcal{F}} x_{ij} & = & 1 & (j\in\mathcal{D}) \\
& x_{ij} & \in & \{0,1\} & (i\in\mathcal{F}, j\in\mathcal{D}) \\
& y_i & \in & \{0,1\} & (i\in\mathcal{F}).
\end{array}$$

En relâchant les contraintes d'intégralité, nous obtenons le PL :

$$\begin{array}{lrcll}
\min & \displaystyle\sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} & & & \\
\text{s.c.} & x_{ij} & \leq & y_i & (i\in\mathcal{F}, j\in\mathcal{D}) \\
& \displaystyle\sum_{i\in\mathcal{F}} x_{ij} & = & 1 & (j\in\mathcal{D}) \\
& x_{ij} & \geq & 0 & (i\in\mathcal{F}, j\in\mathcal{D}) \\
& y_i & \geq & 0 & (i\in\mathcal{F}).
\end{array} \tag{22.2}$$

Ce PL a été formulé en premier par Balinski [1965]. Son dual est le PL suivant :

$$\begin{array}{lrcll}
\max & \displaystyle\sum_{j\in\mathcal{D}} v_j & & & \\
\text{s.c.} & v_j - w_{ij} & \leq & c_{ij} & (i\in\mathcal{F}, j\in\mathcal{D}) \\
& \displaystyle\sum_{j\in\mathcal{D}} w_{ij} & \leq & f_i & (i\in\mathcal{F}) \\
& w_{ij} & \geq & 0 & (i\in\mathcal{F}, j\in\mathcal{D}).
\end{array} \tag{22.3}$$

Les algorithmes d'arrondi résolvent ces PLs (voir théorème 4.18) et arrondissent convenablement la solution fractionnaire du PL primal. Shmoys, Tardos et Aardal [1997] ont obtenu la première approximation de facteur constant à l'aide de cette technique :

ALGORITHME DE SHMOYS-TARDOS-AARDAL

Input Une instance $(\mathcal{D}, \mathcal{F}, (f_i)_{i\in\mathcal{F}}, (c_{ij})_{i\in\mathcal{F}, j\in\mathcal{D}})$ du PROBLÈME DE LOCALISATION SANS CAPACITÉS.

Output Une solution $X \subseteq \mathcal{F}$ et $\sigma : \mathcal{D} \to X$.

① Calculer une solution optimale (x^*, y^*) de (22.2) et une solution optimale (v^*, w^*) de (22.3).

② Poser $k := 1$, $X := \emptyset$ et $U := \mathcal{D}$.

③ Soit $j_k \in U$ tel que $v^*_{j_k}$ soit minimum.
Soit $i_k \in \mathcal{F}$ tel que $x^*_{i_k j_k} > 0$ et que f_{i_k} soit minimum.
Poser $X := X \cup \{i_k\}$.
Soit $N_k := \{j \in U : \exists i \in \mathcal{F} : x^*_{ij_k} > 0, x^*_{ij} > 0\}$.
Poser $\sigma(j) := i_k$ pour tout $j \in N_k$.
Poser $U := U \setminus N_k$.

④ Poser $k := k + 1$.
If $U \neq \emptyset$ **then go to** ③.

Théorème 22.2. (Shmoys, Tardos et Aardal [1997]) *L'algorithme précédent est un algorithme d'approximation de facteur 4 pour le* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS.

Preuve. D'après la condition des écarts complémentaires (corollaire 3.23), $x^*_{ij} > 0$ implique $v^*_j - w^*_{ij} = c_{ij}$ et ainsi $c_{ij} \leq v^*_j$. Alors le coût de fonctionnement pour le client $j \in N_k$ est inférieur ou égal à

$$c_{i_k j} \;\leq\; c_{ij} + c_{ij_k} + c_{i_k j_k} \;\leq\; v^*_j + 2v^*_{j_k} \;\leq\; 3v^*_j,$$

où i est une installation telle que $x^*_{ij} > 0$ et $x^*_{ij_k} > 0$.

Le coût d'installation f_{i_k} peut être borné par

$$f_{i_k} \;\leq\; \sum_{i\in\mathcal{F}} x^*_{ij_k} f_i \;=\; \sum_{i\in\mathcal{F}: x^*_{ij_k}>0} x^*_{ij_k} f_i \;\leq\; \sum_{i\in\mathcal{F}: x^*_{ij_k}>0} y^*_i f_i.$$

Comme $x^*_{ij_k} > 0$ implique $x^*_{ij_{k'}} = 0$ pour $k \neq k'$, le coût d'installation total est inférieur ou égal à $\sum_{i\in\mathcal{F}} y^*_i f_i$.

En faisant la somme, le coût total est inférieur ou égal à $3\sum_{j\in\mathcal{D}} v^*_j + \sum_{i\in\mathcal{F}} y^*_i f_i$, qui est inférieur ou égal à quatre fois la valeur du PL et donc inférieur ou égal à quatre fois l'optimum. □

Le rapport de performance a été amélioré jusqu'à $1,736$ par Chudak et Shmoys [2003] et à $1,582$ par Sviridenko [2002]. Entre-temps, de meilleures garanties de

performance ont été obtenues avec des algorithmes plus simples et plus rapides, qui n'utilisent pas d'algorithme de programmation linéaire comme sous-programme. Ils seront présentés au paragraphe suivant.

22.3 Méthodes primales-duales

Jain et Vazirani [2001] ont proposé un algorithme d'approximation différent. Il s'agit d'un algorithme primal-dual au sens classique : il calcule simultanément des solutions réalisables du primal et du dual (des PLs présentés au paragraphe 22.2). La solution du primal est entière et la garantie de performance sera conséquence de l'approximation des conditions des écarts complémentaires.

L'algorithme consiste à augmenter continuellement toutes les variables duales (en partant de zéro) et à figer v_j lorsque $j \in \mathcal{D}$ est provisoirement connecté. À chaque étape, posons $w_{ij} := \max\{0, v_j - c_{ij}\}$. Initialement toutes les installations sont fermées. Nous ouvrons provisoirement des installations et connectons des clients lorsque les deux événements suivants se produisent :

- $v_j = c_{ij}$ pour une installation provisoirement ouverte i et un client déconnecté j. Alors posons $\sigma(j) := i$ et figons v_j ;
- $\sum_{j \in \mathcal{D}} w_{ij} = f_i$ pour une installation i qui n'est pas (encore) provisoirement ouverte. Alors ouvrir provisoirement i. Pour tous les clients déconnectés $j \in \mathcal{D}$ tels que $v_j \geq c_{ij}$: posons $\sigma(j) := i$ et figons v_j.

Plusieurs événements peuvent se produire en même temps et sont alors traités dans un ordre arbitraire. On continue jusqu'à ce que tous les clients soient connectés.

Soit maintenant V l'ensemble des installations qui sont provisoirement ouvertes et soit E l'ensemble des paires $\{i, i'\}$ d'installations distinctes, provisoirement ouvertes, telles qu'il y ait un client j pour lequel $w_{ij} > 0$ et $w_{i'j} > 0$. Choisissons un ensemble stable maximal X dans le graphe (V, E). Ouvrons les installations de X. Pour $j \in \mathcal{D}$ tel que $\sigma(j) \notin X$, changer $\sigma(j)$ en un voisin ouvert de $\sigma(j)$ dans (V, E).

En fait, X peut être choisi de manière gloutonne pendant que l'on ouvre provisoirement les installations. Alors l'algorithme peut être décrit plus formellement de la manière suivante. Ici Y est l'ensemble des installations qui ne sont pas (encore) provisoirement ouvertes et $\varphi : \mathcal{F} \setminus Y \to X$ affecte une installation ouverte à chaque installation provisoirement ouverte.

ALGORITHME DE JAIN-VAZIRANI	
Input	Une instance $(\mathcal{D}, \mathcal{F}, (f_i)_{i \in \mathcal{F}}, (c_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}})$ du PROBLÈME DE LOCALISATION SANS CAPACITÉS.
Output	Une solution $X \subseteq \mathcal{F}$ et $\sigma : \mathcal{D} \to X$.

① Poser $X := \emptyset$, $Y := \mathcal{F}$ et $U := \mathcal{D}$.

② Poser $t_1 := \min\{c_{ij} : i \in \mathcal{F} \setminus Y, j \in U\}$.
Poser $t_2 := \min\{\tau : \exists i \in Y : \omega(i,\tau) = f_i\}$, où
$\omega(i,\tau) := \sum_{j\in U} \max\{0, \tau - c_{ij}\} + \sum_{j\in\mathcal{D}\setminus U} \max\{0, v_j - c_{ij}\}$.
Poser $t := \min\{t_1, t_2\}$.

③ **For** $i \in \mathcal{F} \setminus Y$ et $j \in U$ tel que $c_{ij} = t$ **do** :
Poser $\sigma(j) := \varphi(i)$, $v_j := t$ et $U := U \setminus \{j\}$.

④ **For** $i \in Y$ tel que $\omega(i,t) = f_i$ **do :**
Poser $Y := Y \setminus \{i\}$.
If il existe $i' \in X$ et $j \in \mathcal{D} \setminus U$ tels que $v_j > c_{ij}$ et $v_j > c_{i'j}$
then poser $\varphi(i) := i'$
else poser $\varphi(i) := i$, $X := X \cup \{i\}$ et $\sigma(j) := i$ pour tout $j \in \mathcal{D} \setminus U$ tel que $v_j > c_{ij}$.
For $j \in U$ tel que $c_{ij} \le t$ **do** :
Poser $\sigma(j) := \varphi(i)$, $v_j := t$ et $U := U \setminus \{j\}$.

⑤ **If** $U \neq \emptyset$ **then go to** ②.

Théorème 22.3. (Jain et Vazirani [2001]) *Pour des instances métriques I, l'*ALGORITHME DE JAIN-VAZIRANI *ouvre un ensemble X d'installations tel que* $3c_F(X) + c_S(X) \le 3\,\mathrm{OPT}(I)$. *En particulier, il s'agit d'un algorithme d'approximation de facteur 3 pour le* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS. *Le temps de calcul de cet algorithme est $O(m \log m)$, où $m = |\mathcal{F}||\mathcal{D}|$.*

Preuve. Observons d'abord que t est non décroissant au cours de l'algorithme.

L'algorithme calcule une solution du primal X et σ et des nombres v_j, $j \in \mathcal{D}$, qui avec les nombres $w_{ij} := \max\{0, v_j - c_{ij}\}$, $i \in \mathcal{F}, j \in \mathcal{D}$, constituent une solution réalisable du PL dual (22.3). Ainsi $\sum_{j\in\mathcal{D}} v_j \le \mathrm{OPT}(I)$. Pour chaque installation ouverte i, tous les clients j tels que $w_{ij} > 0$ sont connectés à i et $f_i = \sum_{j\in\mathcal{D}} w_{ij}$. De plus, nous affirmons que le coût de fonctionnement pour chaque client j est inférieur ou égal à $3(v_j - w_{\sigma(j)j})$.

Nous distinguons deux cas. Si $c_{\sigma(j)j} = v_j - w_{\sigma(j)j}$, cela est clair. Sinon $c_{\sigma(j)j} > v_j$ et $w_{\sigma(j)j} = 0$. Cela signifie que $\varphi(i) \neq i$ lorsque j est connecté à $\varphi(i)$ à l'étape ③ ou ④. Il existe donc une installation (fermée) $i \in \mathcal{F} \setminus (Y \cup X)$ telle que $c_{ij} \le v_j$ et un client j' tel que $w_{ij'} > 0$ et $w_{\sigma(j)j'} > 0$ et ainsi $c_{ij'} = v_{j'} - w_{ij'} < v_{j'}$ et $c_{\sigma(j)j'} = v_{j'} - w_{\sigma(j)j'} < v_{j'}$. Notons que $v_{j'} \le v_j$, car j' est connecté à $\sigma(j)$ avant j. Nous concluons que $c_{\sigma(j)j} \le c_{\sigma(j)j'} + c_{ij'} + c_{ij} < v_{j'} + v_{j'} + v_j \le 3v_j = 3(v_j - w_{\sigma(j)j})$.

Pour le temps de calcul, nous observons que le nombre d'itérations est inférieur ou égal à $|\mathcal{D}| + 1$ puisque au moins un client est supprimé de U à chaque itération, sauf peut être le premier si $f_i = 0$ pour un $i \in \mathcal{F}$. Le temps total pour calculer t_1 à l'étape ② et pour l'étape ③ est $O(m \log m)$ si nous trions tous les c_{ij} à l'avance. Ensuite, remarquons que $t_2 = \min\left\{\frac{t_2^i}{|U_i|} : i \in Y\right\}$, où

$$t_2^i = f_i + \sum_{j\in\mathcal{D}\setminus U : v_j > c_{ij}} (c_{ij} - v_j) + \sum_{j\in U_i} c_{ij}$$

et U_i est l'ensemble des clients déconnectés qui ont un coût de fonctionnement avec i inférieur ou égal à la nouvelle valeur de t. Comme ce nombre est en fait ce que nous voulons calculer, nous procédons de la manière suivante.

Nous mettons à jour t_2, t_2^i et $|U_i|$ ($i \in Y$) tout au long de l'algorithme. Initialement $t_2 = \infty$, $t_2^i = f_i$ et $|U_i| = 0$ pour tout i. Lorsqu'un nouveau client j est connecté et que $v_j > c_{ij}$ pour $i \in Y$, alors t_2^i est réduit de v_j et $|U_i|$ est réduit de 1, ce qui peut également impliquer un changement de t_2. Cependant, nous devons aussi augmenter $|U_i|$ de 1 et augmenter t_2^i de c_{ij} (et éventuellement modifier t_2) lorsque t atteint c_{ij} pour $i \in Y$ et $j \in U$. Cela peut être fait en remplaçant la définition de t_1 à l'étape ② par $t_1 := \min\{c_{ij} : i \in \mathcal{F}, j \in U\}$ et en réalisant ces mises à jour avant l'étape ⑤ pour tout $i \in Y$ et $j \in U$ tel que $c_{ij} = t$. Notons qu'il y a en tout $O(m)$ mises à jour de ce type et que chacune d'entre elles s'exécute en temps constant.

L'instruction «if» de l'étape ④ peut s'implémenter en temps $O(|\mathcal{D}|)$ puisque $i' \in X$, $j \in \mathcal{D} \setminus U$ et $v_j > c_{i'j}$ impliquent $\sigma(j) = i'$. □

Un meilleur algorithme primal-dual a été proposé par Jain *et al.* [2003]. Une des idées est de relâcher les contraintes de réalisabilité des variables duales. Nous interprétons les variables duales comme les budgets des clients, qu'ils utilisent pour payer les coûts de fonctionnement et pour participer aux coûts de construction des installations. Nous ouvrons une installation lorsque les contributions offertes sont suffisantes pour payer le coût de construction. Les clients connectés n'augmentent plus leurs budgets, mais ils peuvent encore en donner une certaine quantité à d'autres installations si elles sont plus proches et si en se reconnectant à celles-ci, les coûts de fonctionnement diminueraient. L'algorithme procède ainsi.

ALGORITHME PAR AJUSTEMENT DUAL

Input Une instance $(\mathcal{D}, \mathcal{F}, (f_i)_{i\in\mathcal{F}}, (c_{ij})_{i\in\mathcal{F}, j\in\mathcal{D}})$ du PROBLÈME DE LOCALISATION SANS CAPACITÉS.

Output Une solution $X \subseteq \mathcal{F}$ et $\sigma : \mathcal{D} \to X$.

① Soient $X := \emptyset$ et $U := \mathcal{D}$.

② Poser $t_1 := \min\{c_{ij} : i \in X, j \in U\}$.
Poser $t_2 := \min\{\tau : \exists i \in \mathcal{F} \setminus X : \omega(i,\tau) = f_i\}$, où
$\omega(i,\tau) := \sum_{j\in U} \max\{0, \tau - c_{ij}\} + \sum_{j\in\mathcal{D}\setminus U} \max\{0, c_{\sigma(j)j} - c_{ij}\}$.
Poser $t := \min\{t_1, t_2\}$.

③ **For** $i \in X$ et $j \in U$ tel que $c_{ij} \leq t$ **do** :
Poser $\sigma(j) := i$, $v_j := t$ et $U := U \setminus \{j\}$.

④ **For** $i \in \mathcal{F} \setminus X$ tel que $\omega(i,t) = f_i$ **do :**
Poser $X := X \cup \{i\}$.
For $j \in \mathcal{D} \setminus U$ tel que $c_{ij} < c_{\sigma(j)j}$ **do** : Poser $\sigma(j) := i$.
For $j \in U$ tel que $c_{ij} < t$ **do** : Poser $\sigma(j) := i$, $v_j := t$ et $U := U \setminus \{j\}$.

⑤ **If** $U \neq \emptyset$ **then go to** ②.

Théorème 22.4. *L'algorithme précédent calcule les nombres v_j, $j \in \mathcal{D}$ et une solution réalisable X, σ de coût inférieur ou égal à $\sum_{j\in\mathcal{D}} v_j$. Son temps d'exécution est $O(|\mathcal{F}|^2|\mathcal{D}|)$.*

Preuve. La première affirmation est évidente. Le temps de calcul peut être obtenu de la même façon qu'avec l'ALGORITHME DE JAIN-VAZIRANI. Cependant, nous devons mettre à jour tous les t_2^i dès qu'un client est reconnecté, c.-à-d. dès qu'une nouvelle installation est ouverte. □

Nous trouverons un nombre γ tel que $\sum_{j\in D} v_j \le \gamma(f_i + \sum_{j\in D} c_{ij})$ pour toutes les paires $(i, D) \in \mathcal{F} \times 2^{\mathcal{D}}$ (c.-à-d. $(\frac{v_j}{\gamma})_{j\in\mathcal{D}}$ est une solution réalisable du PL dual à l'exercice 3). Cela impliquera le rapport de performance γ. Bien entendu, nous devons supposer que les coûts de fonctionnement sont métriques.

Considérons $i \in \mathcal{F}$ et $D \subseteq \mathcal{D}$ tel que $|D| = d$. Numérotons les clients de D dans l'ordre dans lequel ils sont supprimés de U au cours de l'algorithme ; sans perte de généralité $D = \{1, \dots, d\}$. Nous avons $v_1 \le v_2 \le \cdots \le v_d$.

Soit $k \in D$. Remarquons que k est connecté lorsque $t = v_k$ dans l'algorithme et considérons la situation où t devient égal à v_k à l'étape ② pour la première fois. Pour $j = 1, \dots, k-1$ posons

$$r_{j,k} := \begin{cases} c_{i(j,k)j} & \text{si } j \text{ est connecté à } i(j,k) \in \mathcal{F} \text{ à ce moment} \\ v_k & \text{sinon, c.-à-d. si } v_j = v_k \end{cases}.$$

Nous notons les inégalités valides pour ces variables. Tout d'abord, pour $j = 1, \dots, d-2$,

$$r_{j,j+1} \ge r_{j,j+2} \ge \cdots \ge r_{j,d}, \tag{22.4}$$

car les coûts de fonctionnement diminuent si les clients sont reconnectés. Ensuite, pour $k = 1, \dots, d$,

$$\sum_{j=1}^{k-1} \max\{0, r_{j,k} - c_{ij}\} + \sum_{l=k}^{d} \max\{0, v_k - c_{il}\} \le f_i. \tag{22.5}$$

Pour voir cela , nous considérons deux cas. Si $i \in \mathcal{F} \setminus X$ au temps considéré, alors (22.5) est vérifié d'après le choix de t à l'étape ②. Sinon i a été ajouté à X auparavant et au temps considéré $\sum_{j\in U} \max\{0, v_j - c_{ij}\} + \sum_{j\in\mathcal{D}\setminus U} \max\{0, c_{\sigma(j)j} - c_{ij}\} = f_i$. Par la suite, le terme de gauche ne peut que devenir plus petit.

Finalement, pour $1 \le j < k \le d$,

$$v_k \le r_{j,k} + c_{ij} + c_{ik}. \tag{22.6}$$

L'inégalité est évidente si $r_{j,k} = v_k$. Sinon, on peut voir qu'elle est conséquence du choix de t_1 à l'étape ②, en observant que le terme de droite de (22.6) est supérieur ou égal à $c_{i(j,k)k}$ en raison des coûts de service métriques et que l'installation $i(j,k)$ est ouverte au temps considéré.

Pour obtenir un rapport de performance, nous considérons le problème d'optimisation suivant pour $\gamma_F \geq 1$ et $d \in \mathbb{N}$. Comme nous voulons prouver une affirmation pour toutes les instances, nous considérons f_i, c_{ij} et v_j ($j = 1, \ldots, d$) et $r_{j,k}$ ($1 \leq j < k \leq d$) comme des variables :

$$
\begin{array}{lrcll}
\max & \dfrac{\sum_{j=1}^{d} v_j - \gamma_F f_i}{\sum_{j=1}^{d} c_{ij}} & & & \\
\text{s.c.} & v_j & \leq & v_{j+1} & (1 \leq j < d) \\
& r_{j,k} & \geq & r_{j,k+1} & (1 \leq j < k < d) \\
& r_{j,k} + c_{ij} + c_{ik} & \geq & v_k & (1 \leq j < k \leq d) \\
& \sum_{j=1}^{k-1} \max\{0, r_{j,k} - c_{ij}\} + & & & \\
& \quad \sum_{l=k}^{d} \max\{0, v_k - c_{il}\} & \leq & f_i & (1 \leq k \leq d) \\
& \sum_{j=1}^{d} c_{ij} & > & 0 & \\
& f_i & \geq & 0 & \\
& v_j, c_{ij} & \geq & 0 & (1 \leq j \leq d) \\
& r_{j,k} & \geq & 0 & (1 \leq j < k \leq d).
\end{array}
\tag{22.7}
$$

Remarquons que ce problème d'optimisation peut être facilement reformulé comme un PL (exercice 6). Il est souvent désigné sous le nom du *PL révélateur de facteur*. Ses valeurs optimales impliquent des garanties de performance pour l'ALGORITHME PAR AJUSTEMENT DUAL :

Théorème 22.5. *Soit $\gamma_F \geq 1$, et soit γ_S le supremum des valeurs optimales du PL révélateur de facteur (22.7) pour tout $d \in \mathbb{N}$. Soit une instance du* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS *et considérons une solution $X^* \subseteq \mathcal{F}$. Alors le coût de la solution fournie par l'*ALGORITHME PAR AJUSTEMENT DUAL*, sur cette instance, est inférieur ou égal à $\gamma_F c_F(X^*) + \gamma_S c_S(X^*)$.*

Preuve. L'algorithme produit les nombres v_j et implicitement les nombres $r_{j,k}$ pour tout $j, k \in \mathcal{D}$ tels que $v_j \leq v_k$ et $j \neq k$. Pour chaque paire $(i, D) \in \mathcal{F} \times 2^{\mathcal{D}}$, les nombres $f_i, c_{ij}, v_j, r_{j,k}$ vérifient les conditions (22.4), (22.5) et (22.6) et constituent ainsi une solution réalisable de (22.7) à moins que $\sum_{j=1}^{d} c_{ij} = 0$. Alors $\sum_{j=1}^{d} v_j - \gamma_F f_i \leq \gamma_S \sum_{j=1}^{d} c_{ij}$ (cela est une conséquence directe de (22.5) et (22.6) si $c_{ij} = 0$ pour tout $j \in D$). En choisissant $\sigma^* : \mathcal{D} \to X^*$ tel que $c_{\sigma^*(j)j} = \min_{i \in X^*} c_{ij}$ et en sommant sur toutes les paires $(i, \{j \in \mathcal{D} : \sigma^*(j) = i\})$ ($i \in X^*$), nous obtenons

$$\sum_{j \in \mathcal{D}} v_j \leq \gamma_F \sum_{i \in X^*} f_i + \gamma_S \sum_{j \in \mathcal{D}} c_{\sigma^*(j)j} = \gamma_F c_F(X^*) + \gamma_S c_S(X^*).$$

Comme la solution calculée par l'algorithme a un coût total inférieur ou égal à $\sum_{j \in \mathcal{D}} v_j$, cela prouve le théorème. □

Afin d'appliquer ce résultat, nous faisons l'observation suivante :

Lemme 22.6. *Considérons le PL révélateur de facteur (22.7) pour $d \in \mathbb{N}$.*
(a) *Pour $\gamma_F = 1$, l'optimum est inférieur ou égal à 2.*
(b) (Jain *et al.* [2003]) *Pour $\gamma_F = 1,61$, l'optimum est inférieur ou égal à $1,61$.*
(c) (Mahdian, Ye et Zhang [2006]) *Pour $\gamma_F = 1,11$, l'optimum est inférieur ou égal à $1,78$.*

Preuve. Nous prouvons seulement (a). Pour une solution réalisable, nous avons

$$\begin{aligned} d\left(f_i + \sum_{j=1}^{d} c_{ij}\right) &\geq \sum_{k=1}^{d}\left(\sum_{j=1}^{k-1} r_{j,k} + \sum_{l=k}^{d} v_k\right) \\ &\geq \sum_{k=1}^{d} dv_k - (d-1)\sum_{j=1}^{d} c_{ij}, \end{aligned} \tag{22.8}$$

ce qui implique que $d\sum_{j=1}^{d} v_j \leq df_i + (2d-1)\sum_{j=1}^{d} c_{ij}$, c.-à-d. $\sum_{j=1}^{d} v_j \leq f_i + 2\sum_{j=1}^{d} c_{ij}$. □

Les preuves de (b) et (c) sont assez longues et techniques. (a) implique directement que $(\frac{v_j}{2})_{j\in\mathcal{D}}$ est une solution réalisable du dual et que l'ALGORITHME PAR AJUSTEMENT DUAL est un algorithme d'approximation de facteur 2. (b) implique un rapport de performance égal à $1,61$. On peut obtenir de meilleurs résultats en combinant l'ALGORITHME PAR AJUSTEMENT DUAL avec les techniques de réduction d'échelle et d'augmentation gloutonne, qui seront présentées au paragraphe suivant. Le théorème 22.5 et le lemme 22.6 impliquent le résultat immédiat suivant.

Corollaire 22.7. *Soit $(\gamma_F, \gamma_S) \in \{(1,2),(1,61,1,61),(1,11,1,78)\}$. Étant donné une instance du* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS, *considérons une solution $\emptyset \neq X^* \subseteq \mathcal{F}$. Alors le coût de la solution fournie par l'*ALGORITHME PAR AJUSTEMENT DUAL *sur cette instance est inférieur ou égal à $\gamma_F c_F(X^*) + \gamma_S c_S(X^*)$.* □

22.4 Réduction d'échelle et augmentation gloutonne

De nombreux résultats d'approximation sont asymétriques relativement au coût de construction et au coût de fonctionnement. Le coût de fonctionnement peut souvent être réduit en ouvrant des installations supplémentaires. Cela peut en fait être utilisé pour améliorer plusieurs garanties de performance.

Proposition 22.8. *Soient $\emptyset \neq X, X^* \subseteq \mathcal{F}$. Alors $\sum_{i\in X^*}(c_S(X) - c_S(X\cup\{i\})) \geq c_S(X) - c_S(X^*)$.*

En particulier, il existe un $i \in X^*$ tel que $\frac{c_S(X)-c_S(X\cup\{i\})}{f_i} \geq \frac{c_S(X)-c_S(X^*)}{c_F(X^*)}$.

Preuve. Pour $j \in \mathcal{D}$, soit $\sigma(j) \in X$ tel que $c_{\sigma(j)j} = \min_{i \in X} c_{ij}$ et soit $\sigma^*(j) \in X^*$ tel que $c_{\sigma^*(j)j} = \min_{i \in X^*} c_{ij}$. Alors $c_S(X) - c_S(X \cup \{i\}) \geq \sum_{j \in \mathcal{D}: \sigma^*(j)=i} (c_{\sigma(j)j} - c_{ij})$ pour tout $i \in X^*$. En sommant ces inégalités, on obtient le résultat voulu. □

L'**augmentation gloutonne** d'un ensemble X consiste à choisir itérativement un élément $i \in \mathcal{F}$, maximisant $\frac{c_S(X) - c_S(X \cup \{i\})}{f_i}$, que l'on ajoute à X jusqu'à ce que $c_S(X) - c_S(X \cup \{i\}) \leq f_i$ pour tout $i \in \mathcal{F}$. Nous avons besoin du lemme suivant :

Lemme 22.9. (Charikar et Guha [2005]) *Soient $\emptyset \neq X, X^* \subseteq \mathcal{F}$. Appliquons l'augmentation gloutonne à X, ce qui nous fournit un ensemble $Y \supseteq X$. Alors*

$$c_F(Y) + c_S(Y) \leq c_F(X) + c_F(X^*) \ln \left(\max \left\{ 1, \frac{c_S(X) - c_S(X^*)}{c_F(X^*)} \right\} \right) + c_F(X^*) + c_S(X^*).$$

Preuve. Si $c_S(X) \leq c_F(X^*) + c_S(X^*)$, l'inégalité précédente est évidemment vérifiée, même avec X à la place de Y. L'augmentation gloutonne n'augmente jamais le coût.

Sinon, soit $X = X_0, X_1, \ldots, X_k$ la suite des ensembles augmentés, telle que k soit le premier indice pour lequel $c_S(X_k) \leq c_F(X^*) + c_S(X^*)$. En renumérotant les installations, nous pouvons supposer $X_i \setminus X_{i-1} = \{i\}$ $(i = 1, \ldots, k)$. D'après la proposition 22.8,

$$\frac{c_S(X_{i-1}) - c_S(X_i)}{f_i} \geq \frac{c_S(X_{i-1}) - c_S(X^*)}{c_F(X^*)}$$

pour $i = 1, \ldots, k$. Ainsi $f_i \leq c_F(X^*) \frac{c_S(X_{i-1}) - c_S(X_i)}{c_S(X_{i-1}) - c_S(X^*)}$ (remarquons que $c_S(X_{i-1}) > c_S(X^*)$), et

$$c_F(X_k) + c_S(X_k) \leq c_F(X) + c_F(X^*) \sum_{i=1}^{k} \frac{c_S(X_{i-1}) - c_S(X_i)}{c_S(X_{i-1}) - c_S(X^*)} + c_S(X_k).$$

Puisque le terme de droite augmente lorsque $c_S(X_k)$ augmente (la dérivée est égale à $1 - \frac{c_F(X^*)}{c_S(X_{k-1}) - c_S(X^*)} > 0$), le terme de droite ne devient pas plus petit si nous remplaçons $c_S(X_k)$ par $c_F(X^*) + c_S(X^*)$. En utilisant $x - 1 \geq \ln x$ pour $x > 0$, nous obtenons

$$\begin{aligned} c_F(X_k) + c_S(X_k) &\leq c_F(X) + c_F(X^*) \sum_{i=1}^{k} \left(1 - \frac{c_S(X_i) - c_S(X^*)}{c_S(X_{i-1}) - c_S(X^*)}\right) + c_S(X_k) \\ &\leq c_F(X) - c_F(X^*) \sum_{i=1}^{k} \ln \frac{c_S(X_i) - c_S(X^*)}{c_S(X_{i-1}) - c_S(X^*)} + c_S(X_k) \\ &= c_F(X) - c_F(X^*) \ln \frac{c_S(X_k) - c_S(X^*)}{c_S(X) - c_S(X^*)} + c_S(X_k) \\ &= c_F(X) + c_F(X^*) \ln \frac{c_S(X) - c_S(X^*)}{c_F(X^*)} + c_F(X^*) + c_S(X^*). \end{aligned}$$

□

Cela peut être utilisé pour améliorer plusieurs des garanties de performance précédentes. Il est parfois judicieux de combiner l'augmentation gloutonne avec la réduction d'échelle. Nous obtenons le résultat général suivant :

Théorème 22.10. *Supposons qu'il existe des constantes positives $\beta, \gamma_S, \gamma_F$ et un algorithme A qui, pour chaque instance, calcule une solution X telle que $\beta c_F(X) + c_S(X) \leq \gamma_F c_F(X^*) + \gamma_S c_S(X^*)$ pour tout $\emptyset \neq X^* \subseteq \mathcal{F}$. Soit $\delta \geq \frac{1}{\beta}$.*

Alors en normalisant les coûts de construction par δ, en appliquant l'algorithme A à l'instance modifiée et en appliquant l'augmentation gloutonne au résultat respectivement à l'instance de départ, on obtient une solution de coût inférieur ou égal à $\max\{\frac{\gamma_F}{\beta} + \ln(\beta\delta), 1 + \frac{\gamma_S - 1}{\beta\delta}\}$ fois l'optimum.

Preuve. Soit X^* l'ensemble des installations ouvertes d'une solution optimale de l'instance de départ. Nous avons $\beta\delta c_F(X) + c_S(X) \leq \gamma_F \delta c_F(X^*) + \gamma_S c_S(X^*)$. Si $c_S(X) \leq c_S(X^*) + c_F(X^*)$, alors nous avons $\beta\delta(c_F(X) + c_S(X)) \leq \gamma_F \delta c_F(X^*) + \gamma_S c_S(X^*) + (\beta\delta - 1)(c_S(X^*) + c_F(X^*))$, donc X est une solution dont le coût est inférieur ou égal à $\max\{1 + \frac{\gamma_F \delta - 1}{\beta\delta}, 1 + \frac{\gamma_S - 1}{\beta\delta}\}$ fois l'optimum. Remarquons que $1 + \frac{\gamma_F \delta - 1}{\beta\delta} \leq \frac{\gamma_F}{\beta} + \ln(\beta\delta)$ puisque $1 - \frac{1}{x} \leq \ln x$ pour tout $x > 0$.

Sinon, nous appliquons l'augmentation gloutonne à X et nous obtenons une solution de coût inférieur ou égal à

$$\begin{aligned} & c_F(X) + c_F(X^*) \ln \frac{c_S(X) - c_S(X^*)}{c_F(X^*)} + c_F(X^*) + c_S(X^*) \\ \leq\ & c_F(X) + c_F(X^*) \ln \frac{(\gamma_S - 1) c_S(X^*) + \gamma_F \delta c_F(X^*) - \beta\delta c_F(X)}{c_F(X^*)} \\ & + c_F(X^*) + c_S(X^*). \end{aligned}$$

La dérivée de cette expression respectivement à $c_F(X)$ est égale à

$$1 - \frac{\beta\delta c_F(X^*)}{(\gamma_S - 1) c_S(X^*) + \gamma_F \delta c_F(X^*) - \beta\delta c_F(X)}\ .$$

Elle est nulle pour $c_F(X) = \frac{\gamma_F - \beta}{\beta} c_F(X^*) + \frac{\gamma_S - 1}{\beta\delta} c_S(X^*)$. Nous avons ainsi une solution de coût inférieur ou égal à

$$\left(\frac{\gamma_F}{\beta} + \ln(\beta\delta)\right) c_F(X^*) + \left(1 + \frac{\gamma_S - 1}{\beta\delta}\right) c_S(X^*).$$

□

D'après le corollaire 22.7, nous pouvons appliquer ce résultat à l'ALGORITHME PAR AJUSTEMENT DUAL avec $\beta = \gamma_F = 1$ et $\gamma_S = 2$: en prenant $\delta = 1,76$, nous obtenons une garantie de performance égale à $1,57$. Avec $\beta = 1$, $\gamma_F = 1,11$ et $\gamma_S = 1,78$ (voir corollaire 22.7) nous pouvons obtenir encore mieux :

Corollaire 22.11. (Mahdian, Ye et Zhang [2006]) *Multiplions tous les coûts de construction par $\delta = 1,504$, appliquons l'*ALGORITHME PAR AJUSTEMENT DUAL*, normalisons les coûts de construction et appliquons l'augmentation gloutonne. Alors cet algorithme a une garantie de performance égale à $1,52$.* □

Cela est actuellement le meilleur rapport de performance connu pour le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS. Byrka et Aardal [2007] ont montré que le rapport de performance de cet algorithme ne peut pas être meilleur que 1,494. Récemment, Byrka [2007] a amélioré le rapport de performance jusqu'à 1,500.

Dans le cas particulier où les coûts de fonctionnement sont compris entre 1 et 3, l'augmentation gloutonne fournit un meilleur rapport de performance. Soit α la solution de l'équation $\alpha + 1 = \ln \frac{2}{\alpha}$. Nous avons $0,463 \leq \alpha \leq 0,4631$. Un simple calcul montre que $\alpha = \frac{\alpha}{\alpha+1} \ln \frac{2}{\alpha} = \max\{\frac{\xi}{\xi+1} \ln \frac{2}{\xi} : \xi > 0\}$.

Théorème 22.12. (Guha et Khuller [1999]) *Considérons le* PROBLÈME DE LOCALISATION SANS CAPACITÉS *restreint aux instances telles que tous les coûts de fonctionnement sont dans l'intervalle $[1, 3]$. Ce problème a un algorithme d'approximation de facteur $(1 + \alpha + \epsilon)$ pour tout $\epsilon > 0$.*

Preuve. Soit $\epsilon > 0$ et soit $k := \lceil \frac{1}{\epsilon} \rceil$. Énumérons toutes les solutions $X \subseteq \mathcal{F}$ telles que $|X| \leq k$.

Nous calculons une autre solution de la manière suivante. Nous ouvrons d'abord une installation i de coût de construction f_i et appliquons ensuite l'augmentation gloutonne pour obtenir une solution Y. Nous affirmons que le coût de la meilleure solution est inférieur ou égal à $1 + \alpha + \epsilon$ fois l'optimum.

Soient X^* une solution optimale et $\xi = \frac{c_F(X^*)}{c_S(X^*)}$. Nous pouvons supposer que $|X^*| > k$, car sinon nous avons trouvé X^* précédemment. Alors $c_F(\{i\}) \leq \frac{1}{k} c_F(X^*)$. De plus, comme les coûts de fonctionnement sont compris entre 1 et 3, $c_S(\{i\}) \leq 3|\mathcal{D}| \leq 3c_S(X^*)$.

D'après le lemme 22.9, le coût de Y est inférieur ou égal à

$$
\begin{aligned}
& \frac{1}{k}c_F(X^*) + c_F(X^*)\ln\left(\max\left\{1, \frac{2c_S(X^*)}{c_F(X^*)}\right\}\right) + c_F(X^*) + c_S(X^*) \\
= \;& c_S(X^*)\left(\frac{\xi}{k} + \xi\ln\left(\max\left\{1, \frac{2}{\xi}\right\}\right) + \xi + 1\right) \\
\leq \;& c_S(X^*)(1+\xi)\left(1 + \epsilon + \frac{\xi}{\xi+1}\ln\left(\max\left\{1, \frac{2}{\xi}\right\}\right)\right) \\
\leq \;& (1+\alpha+\epsilon)(1+\xi)c_S(X^*) \\
= \;& (1+\alpha+\epsilon)(c_F(X^*) + c_S(X^*)).
\end{aligned}
$$

□

Cette garantie de performance semble être la meilleure possible pour la raison suivante :

Théorème 22.13. *S'il existe un $\epsilon > 0$ et un algorithme d'approximation de facteur $(1+\alpha-\epsilon)$ pour le* PROBLÈME DE LOCALISATION SANS CAPACITÉS *restreint aux instances telles que les coûts de fonctionnement sont seulement égaux à 1 et 3, alors $P = NP$.*

Ce théorème a été prouvé par Sviridenko [non publié] (il est fondé sur les résultats de Feige [1998] et Guha et Khuller [1999]) et peut être trouvé dans l'étude de Vygen [2005].

22.5 Bornes du nombre d'installations

Le PROBLÈME DE LOCALISATION DE k INSTALLATIONS est le PROBLÈME DE LOCALISATION SANS CAPACITÉS avec la contrainte supplémentaire que l'on ne peut pas ouvrir plus de k installations, où k est un nombre entier qui fait partie de l'instance. Un cas particulier, où les coûts de construction des installations sont nuls, est le PROBLÈME DU k-MÉDIAN bien connu. Dans ce paragraphe, nous décrivons un algorithme d'approximation pour le PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS.

Pour les problèmes qui deviennent plus simples si un certain type de contraintes est omis, la relaxation lagrangienne (voir paragraphe 5.6) est une technique courante. Ici nous relâchons la borne du nombre d'installations que l'on peut ouvrir et nous ajoutons une constante λ à chaque coût de construction d'une installation.

Théorème 22.14. (Jain et Vazirani [2001]) *S'il existe une constante γ_S et un algorithme polynomial A tels que, pour toute instance du* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS*, l'algorithme A calcule une solution X telle que $c_F(X) + c_S(X) \leq c_F(X^*) + \gamma_S c_S(X^*)$ pour tout $\emptyset \neq X^* \subseteq \mathcal{F}$, alors il existe un algorithme d'approximation de facteur $(2\gamma_S)$ pour le* PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS *avec des données entières.*

Preuve. Considérons une instance du PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS. Nous supposons que les coûts de fonctionnement sont des entiers dans $\{0, 1, \ldots, c_{\max}\}$ et que les coûts de construction des installations sont des entiers dans $\{0, 1, \ldots, f_{\max}\}$.

Nous vérifions d'abord s'il existe une solution de coût nul et en trouvons une si c'est le cas. Cela est facile (voir la preuve du lemme 22.15). Ainsi, nous supposons que le coût d'une solution est supérieur ou égal à 1. Soit X^* une solution optimale (nous ne l'utiliserons que pour l'analyse du problème).

Soit $A(\lambda) \subseteq \mathcal{F}$ la solution calculée par A pour l'instance où tous les coûts de construction des installations sont augmentés de λ, mais où la contrainte sur le nombre d'installations est omise. Nous avons $c_F(A(\lambda)) + |A(\lambda)|\lambda + c_S(A(\lambda)) \leq c_F(X^*) + |X^*|\lambda + \gamma_S c_S(X^*)$ et ainsi

$$c_F(A(\lambda)) + c_S(A(\lambda)) \leq c_F(X^*) + \gamma_S c_S(X^*) + (k - |A(\lambda)|)\lambda \tag{22.9}$$

pour tout $\lambda \geq 0$. Si $|A(0)| \leq k$, alors $A(0)$ est une solution réalisable coûtant au plus γ_S fois l'optimum et nous avons terminé.

Sinon $|A(0)| > k$ et remarquons que $|A(f_{\max} + \gamma_S|\mathcal{D}|c_{\max} + 1)| = 1 \leq k$. Posons $\lambda' := 0$ et $\lambda'' := f_{\max} + \gamma_S|\mathcal{D}|c_{\max} + 1$ et appliquons une recherche dichotomique où on maintient $|A(\lambda'')| \leq k < |A(\lambda')|$. Après $O(\log|\mathcal{D}| + \log f_{\max} + \log c_{\max})$ itérations, telles qu'à chacune d'entre elles nous remplaçons λ' ou λ'' par leur moyenne arithmétique selon que $\left|A\left(\frac{\lambda'+\lambda''}{2}\right)\right| \leq k$ ou non, nous avons $\lambda'' - \lambda' \leq \frac{1}{|\mathcal{D}|^2}$. (Remarquons que cette recherche dichotomique fonctionne, bien que $\lambda \mapsto |A(\lambda)|$ ne soit pas monotone en général.)

Si $|A(\lambda'')| = k$, alors (22.9) implique que $A(\lambda'')$ est une solution réalisable coûtant au plus γ_S fois l'optimum et nous avons terminé. Cependant, nous ne rencontrerons pas toujours un tel λ'', car $\lambda \mapsto |A(\lambda)|$ n'est pas toujours monotone et peut varier de plus de 1 (Archer, Rajagopalan et Shmoys [2003] ont montré comment arranger cela à l'aide de coûts perturbants, mais ils n'ont pas été en mesure de le faire en temps polynomial).

Nous considérons ainsi $X := A(\lambda')$ et $Y := A(\lambda'')$ et nous supposons désormais $|X| > k > |Y|$. Soient $\alpha := \frac{k-|Y|}{|X|-|Y|}$ et $\beta := \frac{|X|-k}{|X|-|Y|}$.

Choisissons un sous-ensemble X' de X de telle sorte que $|X'| = |Y|$ et tel que $\min_{i\in X'} c_{ii'} = \min_{i\in X} c_{ii'}$ pour chaque $i' \in Y$, où $c_{ii'} := \min_{j\in\mathcal{D}}(c_{ij} + c_{i'j})$.

Nous ouvrons soit tous les éléments de X' (avec probabilité α), soit tous les éléments de Y (avec probabilité $\beta = 1 - \alpha$). De plus, nous ouvrons un ensemble de $k - |Y|$ installations de $X \setminus X'$, choisies de façon aléatoire et uniforme. Alors l'espérance du coût de construction est égale à $\alpha c_F(X) + \beta c_F(Y)$. (Remarquons que X et Y ne sont pas nécessairement disjoints et nous pouvons même payer deux fois l'ouverture de certaines installations. Ainsi $\alpha c_F(X) + \beta c_F(Y)$ est en fait une borne supérieure de l'espérance du coût de construction.)

Soit $j \in \mathcal{D}$ et soit i' une plus proche installation dans X et i'' une plus proche installation dans Y. Connectons j à i' si i' est ouverte, ou à i'' si i'' est ouverte. Si ni

i' ni i'' ne sont ouvertes, nous connectons j à une installation $i''' \in X'$ minimisant $c_{i''i'''}$.

Cela fournit une espérance du coût de fonctionnement égale à $\alpha c_{i'j} + \beta c_{i''j}$ si $i' \in X'$ et inférieure ou égale à

$$\begin{aligned} & \alpha c_{i'j} + (1-\alpha)\beta c_{i''j} + (1-\alpha)(1-\beta)c_{i'''j} \\ \leq\ & \alpha c_{i'j} + \beta^2 c_{i''j} + \alpha\beta\left(c_{i''j} + \min_{j' \in \mathcal{D}}(c_{i''j'} + c_{i'''j'})\right) \\ \leq\ & \alpha c_{i'j} + \beta^2 c_{i''j} + \alpha\beta(c_{i''j} + c_{i''j} + c_{i'j}) \\ =\ & \alpha(1+\beta)c_{i'j} + \beta(1+\alpha)c_{i''j} \end{aligned}$$

si $i' \in X \setminus X'$.

Ainsi l'espérance totale du coût de fonctionnement est inférieure ou égale à

$$(1 + \max\{\alpha, \beta\})(\alpha c_S(X) + \beta c_S(Y)) \ \leq\ \left(2 - \frac{1}{|\mathcal{D}|}\right)(\alpha c_S(X) + \beta c_S(Y)).$$

En utilisant (22.9), nous obtenons en tout une espérance de coût inférieure ou égale à

$$\begin{aligned} & \left(2 - \frac{1}{|\mathcal{D}|}\right)\left(\alpha(c_F(X) + c_S(X)) + \beta(c_F(Y) + c_S(Y))\right) \\ \leq\ & \left(2 - \frac{1}{|\mathcal{D}|}\right)\left(c_F(X^*) + \gamma_S c_S(X^*) + (\lambda'' - \lambda')\frac{(|X| - k)(k - |Y|)}{|X| - |Y|}\right) \\ \leq\ & \left(2 - \frac{1}{|\mathcal{D}|}\right)\left(c_F(X^*) + \gamma_S c_S(X^*) + (\lambda'' - \lambda')\frac{|X| - |Y|}{4}\right) \\ \leq\ & \left(2 - \frac{1}{|\mathcal{D}|}\right)\left(c_F(X^*) + \gamma_S c_S(X^*) + \frac{1}{4|\mathcal{D}|}\right) \\ \leq\ & \left(2 - \frac{1}{|\mathcal{D}|}\right)\left(1 + \frac{1}{4|\mathcal{D}|}\right)(c_F(X^*) + \gamma_S c_S(X^*)) \\ \leq\ & \left(2 - \frac{1}{2|\mathcal{D}|}\right)(c_F(X^*) + \gamma_S c_S(X^*)) \end{aligned}$$

et ainsi inférieure ou égale à $2\gamma_S(c_F(X^*) + c_S(X^*))$.

Remarquons que l'espérance du coût est facile à calculer même sous la condition qu'un sous-ensemble Z est ouvert avec probabilité 1 et que $k - |Z|$ installations, choisies aléatoirement dans un autre ensemble, sont ouvertes. On peut alors dérandomiser cet algorithme par la méthode des probabilités conditionnelles : tout d'abord ouvrons X' ou Y selon que la borne de l'espérance du coût est inférieure ou égale à $\left(2 - \frac{1}{|\mathcal{D}|}\right)(\alpha(c_F(X) + c_S(X)) + \beta(c_F(Y) + c_S(Y)))$ ou non et ensuite ouvrons successivement les installations de $X \setminus X'$ de telle façon que la borne soit encore vérifiée. □

En particulier, nous obtenons grâce à l'ALGORITHME PAR AJUSTEMENT DUAL (corollaire 22.7), un algorithme d'approximation de facteur 4 pour le PROBLÈME

DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS avec des données entières. Le premier algorithme d'approximation de facteur constant pour le PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS était dû à Charikar *et al.* [2002].

Le temps de calcul de la recherche dichotomique est faiblement polynomial et cela n'est vrai que pour des données entières. Cependant, nous pouvons le rendre fortement polynomial en discrétisant les données de l'input :

Lemme 22.15. *Pour toute instance I du* PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS, *$\gamma_{\max} \geq 1$ et $0 < \epsilon \leq 1$, nous pouvons décider si $\mathrm{OPT}(I) = 0$ et sinon générer une autre instance I' en temps $O(|\mathcal{F}||\mathcal{D}| \log(|\mathcal{F}||\mathcal{D}|))$, telle que tous les coûts de construction et de fonctionnement soient des entiers dans $\left\{0, 1, \ldots, \left\lceil \frac{2\gamma_{\max}(k+|\mathcal{D}|)^3}{\epsilon} \right\rceil\right\}$ et que, pour tout $1 \leq \gamma \leq \gamma_{\max}$, toute solution de I' de coût inférieur ou égal à $\gamma\,\mathrm{OPT}(I')$, soit une solution de I de coût inférieur ou égal à $\gamma(1+\epsilon)\,\mathrm{OPT}(I)$.*

Preuve. Soit $n := k + |\mathcal{D}|$. Étant donné une instance I, nous calculons d'abord une borne supérieure et une borne inférieure de $\mathrm{OPT}(I)$ différant d'un facteur au plus égal à $2n^2 - 1$, de la manière suivante. Pour chaque $B \in \{f_i : i \in \mathcal{F}\} \cup \{c_{ij} : i \in \mathcal{F}, j \in \mathcal{D}\}$, nous considérons le graphe biparti $G_B := (\mathcal{D} \cup \mathcal{F}, \{(i,j) : i \in \mathcal{F}, j \in \mathcal{D}, f_i \leq B, c_{ij} \leq B\})$.

La plus petite valeur de B, telle que les éléments de $\mathcal{D}$ appartiennent à au plus k composantes connexes différentes de G_B, chacune d'entre elles contenant au moins une installation, est une borne inférieure de $\mathrm{OPT}(I)$. Ce nombre B peut être trouvé en temps $O(|\mathcal{F}||\mathcal{D}| \log(|\mathcal{F}||\mathcal{D}|))$ à l'aide d'une simple variante de l'ALGORITHME DE KRUSKAL pour les arbres couvrants minimaux.

De plus, pour cette valeur de B, on peut choisir une installation arbitraire dans chaque composante connexe de G_B, qui contient un élément de $\mathcal{D}$ et connecter chaque client de telle manière que le coût de fonctionnement soit inférieur ou égal à $(2|\mathcal{D}| - 1)B$ (en utilisant l'hypothèse que les coûts de fonctionnement sont métriques). Ainsi $\mathrm{OPT}(I) \leq kB + (2|\mathcal{D}| - 1)|\mathcal{D}|B < (2n^2 - 1)B$ à moins que $B = 0$, auquel cas nous avons terminé.

Nous pouvons ainsi ignorer les coûts de construction et de fonctionnement excédant $B' := 2\gamma_{\max} n^2 B$. Nous obtenons I' à partir de I en fixant chaque c_{ij} à $\left\lceil \frac{\min\{B', c_{ij}\}}{\delta} \right\rceil$ et chaque f_i à $\left\lceil \frac{\min\{B', f_i\}}{\delta} \right\rceil$, où $\delta = \frac{\epsilon B}{n^3}$. Tous les nombres de l'input sont maintenant des entiers dans $\left\{0, 1, \ldots, \left\lceil \frac{2\gamma_{\max} n^3}{\epsilon} \right\rceil\right\}$.

Nous avons

$$\mathrm{OPT}(I') \leq \frac{\mathrm{OPT}(I)}{\delta} + n = \frac{\mathrm{OPT}(I) + \epsilon B}{\delta} < \frac{(2n^2 - 1)B + \epsilon B}{\delta} \leq \frac{2n^2 B}{\delta} = \frac{B'}{\gamma_{\max}\delta}$$

et ainsi une solution de I', de coût inférieur ou égal à $\gamma\,\mathrm{OPT}(I')$, ne contient pas d'élément de coût $\lceil \frac{B'}{\delta} \rceil$ et est ainsi une solution de I de coût inférieur ou égal à

$$\delta\gamma\,\mathrm{OPT}(I') \leq \gamma(\mathrm{OPT}(I) + \epsilon B) \leq \gamma(1+\epsilon)\,\mathrm{OPT}(I).$$

□

Corollaire 22.16. *Il existe un algorithme d'approximation de facteur* 4 *fortement polynomial pour le* PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS.

Preuve. Appliquons le lemme 22.15 avec $\gamma_{\max} = 4$ et $\epsilon = \frac{1}{4|\mathcal{D}|}$ et appliquons le théorème 22.14 avec l'ALGORITHME PAR AJUSTEMENT DUAL à l'instance obtenue. Nous avons $\gamma_S = 2$ d'après le corollaire 22.7 et nous obtenons une solution de coût total inférieur ou égal à

$$\left(2 - \frac{1}{2|\mathcal{D}|}\right)\left(1 + \frac{1}{4|\mathcal{D}|}\right)(c_F(X^*) + \gamma_S c_S(X^*)) \leq 4\,(c_F(X^*) + c_S(X^*))$$

pour tout $\emptyset \neq X^* \subseteq \mathcal{F}$. □

Zhang [2007] a trouvé un algorithme d'approximation de facteur 3,733 pour le PROBLÈME DE LOCALISATION MÉTRIQUE DE k INSTALLATIONS. Cet algorithme utilise des techniques de recherche locale similaires à celles présentées au paragraphe suivant.

22.6 Recherche locale

Comme nous l'avons déjà vu au paragraphe 21.3, la recherche locale est une technique qui est souvent appliquée avec succès en pratique, bien que l'on ne puisse généralement pas prouver de bonnes garanties de performance. Il fut ainsi surprenant de constater que les problèmes de localisation peuvent être bien approximés à l'aide de la recherche locale. Cela a été en premier étudié par Korupolu, Plaxton et Rajaraman [2000] et a mené successivement à plusieurs résultats importants. Nous allons présenter certains d'entre eux dans ce paragraphe et dans le suivant.

Pour le PROBLÈME DU k-MÉDIAN MÉTRIQUE, la recherche locale a fourni le meilleur rapport de performance connu. Avant de présenter ce résultat, nous considérons le plus simple algorithme de recherche locale possible : nous partons d'une solution réalisable quelconque (c.-à-d. un ensemble de k installations) et l'améliorons par des échanges simples. Remarquons que nous ne devons considérer que les coûts de fonctionnement, puisque les coûts de construction sont nuls dans le PROBLÈME DU k-MÉDIAN. De plus, nous pouvons sans perte de généralité supposer qu'une solution doit contenir exactement k installations.

Théorème 22.17. (Arya *et al.* [2004]) *Considérons une instance du* PROBLÈME DU k-MÉDIAN MÉTRIQUE. *Soient X une solution réalisable et X^* une solution optimale. Si $c_S((X \setminus \{x\}) \cup \{y\}) \geq c_S(X)$ pour tout $x \in X$ et $y \in X^*$, alors $c_S(X) \leq 5c_S(X^*)$.*

Preuve. Considérons des affectations optimales σ et σ^* des clients aux k installations dans X et X^*, respectivement. On dit que $x \in X$ *capture* $y \in X^*$ si $|\{j \in \mathcal{D} : \sigma(j) = x, \sigma^*(j) = y\}| > \frac{1}{2}|\{j \in \mathcal{D} : \sigma^*(j) = y\}|$. Chaque $y \in X^*$ est capturé par au plus un $x \in X$.

Soit $\pi : \mathcal{D} \to \mathcal{D}$ une bijection telle que pour tout $j \in \mathcal{D}$:

- $\sigma^*(\pi(j)) = \sigma^*(j)$; et
- si $\sigma(\pi(j)) = \sigma(j)$ alors $\sigma(j)$ capture $\sigma^*(j)$.

Une telle application π peut être obtenue facilement en ordonnant, pour chaque $y \in X^*$, les éléments de $\{j \in \mathcal{D} : \sigma^*(j) = y\} = \{j_0, \ldots, j_{t-1}\}$ de telle façon que les clients j de même valeur $\sigma(j)$ soient consécutifs et en posant $\pi(j_k) := j_{k'}$, où $k' = (k + \lfloor \frac{t}{2} \rfloor) \bmod t$.

Définissons un *échange* par un élément de $X \times X^*$. Pour un échange (x, y), nous appelons x la source et y la cible. Nous allons définir k échanges tels que chaque $y \in X^*$ soit la cible d'exactement l'un d'entre eux.

Si un $x \in X$ capture seulement une installation $y \in X^*$, nous considérons un échange (x, y). S'il existe l tels échanges, alors il reste $k-l$ éléments dans X et dans X^*. Certains des éléments restants de X (au plus $\frac{k-l}{2}$) peuvent capturer au moins deux installations de X^* ; nous ne les considérerons pas. Pour chaque installation restante $y \in X^*$, nous choisissons un $x \in X$ de telle sorte que x ne capture aucune installation et tel que chaque $x \in X$ soit la source d'au plus deux tels échanges.

Nous analysons maintenant les échanges un par un. Considérons l'échange (x, y) et posons $X' := (X \setminus \{x\}) \cup \{y\}$. Alors $c_S(X') \geq c_S(X)$. Transformons $\sigma : \mathcal{D} \to X$ en une nouvelle affectation $\sigma' : \mathcal{D} \to X'$, en réaffectant les clients de la manière suivante.

Les clients $j \in \mathcal{D}$ tels que $\sigma^*(j) = y$ sont affectés à y. Les clients $j \in \mathcal{D}$ tels que $\sigma(j) = x$ et $\sigma^*(j) = y' \in X^* \setminus \{y\}$ sont affectés à $\sigma(\pi(j))$. Remarquons que $\sigma(\pi(j)) \neq x$ puisque x ne capture pas y'. Pour tous les autres clients, l'affectation n'est pas modifiée.

Nous avons

$$\begin{aligned}
0 &\leq c_S(X') - c_S(X) \\
&\leq \sum_{j \in \mathcal{D}:\sigma^*(j)=y} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D}:\sigma(j)=x, \sigma^*(j)\neq y} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\
&\leq \sum_{j \in \mathcal{D}:\sigma^*(j)=y} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D}:\sigma(j)=x} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j})
\end{aligned}$$

puisque $c_{\sigma(\pi(j))j} \geq \min_{i \in X} c_{ij} = c_{\sigma(j)j}$ par définition de σ.

Nous faisons maintenant la somme sur tous les échanges. Remarquons que chaque installation de X^* est la cible d'exactement un échange, donc la somme des premiers termes est égale à $c_S(X^*) - c_S(X)$. De plus, chaque $x \in X$ est la source d'au plus deux échanges. Alors

$$\begin{aligned}
0 &\leq \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + 2 \sum_{j \in \mathcal{D}} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\
&\leq c_S(X^*) - c_S(X) + 2 \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} + c_{\sigma^*(j)\pi(j)} + c_{\sigma(\pi(j))\pi(j)} - c_{\sigma(j)j}) \\
&= c_S(X^*) - c_S(X) + 2 \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} + c_{\sigma^*(\pi(j))\pi(j)}) \\
&= c_S(X^*) - c_S(X) + 4c_S(X^*),
\end{aligned}$$

car π est une bijection. □

Ainsi un optimum local est une approximation de facteur 5. Cependant, cela ne donne aucun résultat en ce qui concerne le temps de calcul nécessaire pour atteindre l'optimum local. En théorie, le nombre d'étapes nécessaires pour atteindre un optimum local pourrait être exponentiel. Cependant, en discrétisant les coûts, nous obtenons un temps de calcul fortement polynomial :

Corollaire 22.18. *Soit $0 < \epsilon \leq 1$. Alors l'algorithme suivant est un algorithme d'approximation de facteur $(5+\epsilon)$ fortement polynomial pour le* PROBLÈME DU k-MÉDIAN MÉTRIQUE *: transformer l'instance comme au lemme 22.15 avec $\gamma_{\max} = 5$ et $\frac{\epsilon}{5}$ à la place de ϵ, commencer avec un ensemble quelconque de k installations et appliquer les échanges qui permettent de diminuer les coûts de fonctionnement tant que cela est possible.*

Preuve. Puisque chacun des coûts de fonctionnement de la nouvelle instance est un entier de l'ensemble $\left\{0, 1, \ldots, \left\lceil \frac{50(k+|\mathcal{D}|)^3}{\epsilon} \right\rceil \right\}$, on peut appliquer au plus $|\mathcal{D}| \left\lceil \frac{50(k+|\mathcal{D}|)^3}{\epsilon} \right\rceil$ échanges successifs réduisant le coût de fonctionnement total. □

On peut améliorer le rapport de performance de manière significative en utilisant des multiéchanges :

Théorème 22.19. (Arya *et al.* [2004]) *Considérons une instance du* PROBLÈME DU k-MÉDIAN MÉTRIQUE *et soit $p \in \mathbb{N}$. Soient X une solution réalisable et X^* une solution optimale. Si $c_S((X \setminus A) \cup B) \geq c_S(X)$ pour tout $A \subseteq X$ et $B \subseteq X^*$ tel que $|A| = |B| \leq p$, alors $c_S(X) \leq (3 + \frac{2}{p}) c_S(X^*)$.*

Preuve. Soit de nouveau σ et σ^* des affectations optimales des clients aux k installations dans X et X^*, respectivement. Pour chaque $A \subseteq X$, soit $C(A)$ l'ensemble des installations dans X^* qui sont capturées par A, c.-à-d.

$$C(A) := \left\{ y \in X^* : |\{j \in \mathcal{D} : \sigma(j) \in A, \sigma^*(j) = y\}| > \frac{1}{2} |\{j \in \mathcal{D} : \sigma^*(j) = y\}| \right\}.$$

Nous partitionnons $X = A_1 \dot\cup \cdots \dot\cup A_r$ et $X^* = B_1 \dot\cup \cdots \dot\cup B_r$ de la manière suivante :

Soit $\{x \in X : C(\{x\}) \neq \emptyset\} =: \{x_1, \ldots, x_s\} =: \bar{X}$.
Poser $r := \max\{s, 1\}$.
For $i = 1$ **to** $r - 1$ **do** :
  Poser $A_i := \{x_i\}$.
  While $|A_i| < |C(A_i)|$ **do :**
    Ajouter un élément $x \in X \setminus (A_1 \cup \cdots \cup A_i \cup \bar{X})$ à A_i.
  Poser $B_i := C(A_i)$.
Poser $A_r := X \setminus (A_1 \cup \cdots \cup A_{r-1})$ et $B_r := X^* \setminus (B_1 \cup \cdots \cup B_{r-1})$.

Il est clair que cet algorithme garantit que $|A_i| = |B_i| \geq 1$ pour $i = 1, \ldots, r$, que les ensembles $A_1, \ldots, A_r$ sont deux à deux disjoints et que les ensembles

$B_1, \ldots, B_r$ sont également deux à deux disjoints. Notons qu'il est toujours possible d'ajouter un élément si $|A_i| < |C(A_i)|$, car alors

$$\begin{aligned} & |X \setminus (A_1 \cup \cdots \cup A_i \cup \bar{X})| \\ = \; & |X| - |A_1| - \cdots - |A_i| - |\{x_{i+1}, \ldots, x_r\}| \\ > \; & |X^*| - |C(A_1)| - \cdots - |C(A_i)| - |C(\{x_{i+1}\})| - \cdots - |C(\{x_r\})| \\ = \; & |X^* \setminus (C(A_1) \cup \cdots \cup C(A_i) \cup C(\{x_{i+1}\}) \cup \cdots \cup C(\{x_r\}))| \\ \geq \; & 0. \end{aligned}$$

Soit $\pi : \mathcal{D} \to \mathcal{D}$ une bijection telle que pour tout $j \in \mathcal{D}$:

- $\sigma^*(\pi(j)) = \sigma^*(j)$;
- si $\sigma(\pi(j)) = \sigma(j)$, alors $\sigma(j)$ capture $\sigma^*(j)$; et
- si $\sigma(j) \in A_i$ et $\sigma(\pi(j)) \in A_i$ pour un certain $i \in \{1, \ldots, r\}$, alors A_i capture $\sigma^*(j)$.

Une telle application π peut être obtenue presque de la même façon que dans la preuve du théorème 22.17.

Nous définissons maintenant un ensemble d'échanges (A, B) tels que $|A| = |B| \leq p$, $A \subseteq X$ et $B \subseteq X^*$. Chaque échange sera associé à un poids positif. L'échange (A, B) signifie que X est remplacé par $X' := (X \setminus A) \cup B$. On dit que A est l'ensemble source et que B est l'ensemble cible.

Pour chaque $i \in \{1, \ldots, r\}$ tel que $|A_i| \leq p$, nous considérons l'échange (A_i, B_i) de poids 1. Pour chaque $i \in \{1, \ldots, r\}$ tel que $|A_i| = q > p$, nous considérons l'échange $(\{x\}, \{y\})$ de poids $\frac{1}{q-1}$ pour chaque $x \in A_i \setminus \{x_i\}$ et chaque $y \in B_i$. Chaque $y \in X^*$ appartient à l'ensemble cible des échanges de poids total égal à 1 tandis que chaque $x \in X$ appartient à l'ensemble source des échanges de poids total inférieur ou égal à $\frac{p+1}{p}$.

Nous réaffectons les clients comme dans le cas des échanges simples. Plus précisément, pour un échange (A, B) nous réaffectons tous les $j \in \mathcal{D}$, tels que $\sigma^*(j) \in B$, à $\sigma^*(j)$ et tous les $j \in \mathcal{D}$, tels que $\sigma^*(j) \notin B$ et $\sigma(j) \in A$, à $\sigma(\pi(j))$. Notons que nous avons $B \supseteq C(A)$ pour chacun des échanges considérés (A, B). Ainsi, pour tout $j \in \mathcal{D}$ tel que $\sigma(j) \in A$ et $\sigma^*(j) \notin B$, nous avons $\sigma(\pi(j)) \notin A$. Alors, nous pouvons borner l'augmentation du coût due à l'échange de la manière suivante :

$$\begin{aligned} 0 \; \leq \; & c_S(X') - c_S(X) \\ \leq \; & \sum_{j \in \mathcal{D} : \sigma^*(j) \in B} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D} : \sigma(j) \in A, \sigma^*(j) \notin B} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\ \leq \; & \sum_{j \in \mathcal{D} : \sigma^*(j) \in B} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D} : \sigma(j) \in A} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}), \end{aligned}$$

car $c_{\sigma(\pi(j))j} \geq c_{\sigma(j)j}$ par définition de σ. Ainsi en prenant la somme pondérée sur tous les échanges, on obtient

$$\begin{aligned}
0 &\leq \sum_{j\in\mathcal{D}}(c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \frac{p+1}{p}\sum_{j\in\mathcal{D}}(c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\
&\leq c_S(X^*) - c_S(X) + \frac{p+1}{p}\sum_{j\in\mathcal{D}}(c_{\sigma^*(j)j} + c_{\sigma^*(j)\pi(j)} + c_{\sigma(\pi(j))\pi(j)} - c_{\sigma(j)j}) \\
&= c_S(X^*) - c_S(X) + \frac{p+1}{p}\sum_{j\in\mathcal{D}}(c_{\sigma^*(j)j} + c_{\sigma^*(\pi(j))\pi(j)}) \\
&= c_S(X^*) - c_S(X) + 2\,\frac{p+1}{p}c_S(X^*),
\end{aligned}$$

car π est une bijection. □

Arya *et al.* [2004] ont également montré que cette garantie de performance est serrée. De même que le corollaire 22.18, le lemme 22.15 et le théorème 22.19 impliquent un algorithme d'approximation de facteur $(3+\epsilon)$ pour tout $\epsilon > 0$. Cela est le meilleur rapport de performance actuellement connu pour le PROBLÈME DU k-MÉDIAN MÉTRIQUE.

Nous pouvons appliquer des techniques similaires pour le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS et obtenir un simple algorithme d'approximation fondé sur la recherche locale :

Théorème 22.20. (Arya *et al.* [2004]) *Considérons une instance du* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS. *Soient X et X^* des solutions réalisables. Si ni $X \setminus \{x\}$ ni $X \cup \{y\}$ ni $(X \setminus \{x\}) \cup \{y\}$ ne sont meilleurs que X pour tout $x \in X$ et pour tout $y \in \mathcal{F} \setminus X$, alors $c_S(X) \leq c_F(X^*) + c_S(X^*)$ et $c_F(X) \leq c_F(X^*) + 2c_S(X^*)$.*

Preuve. Nous utilisons les mêmes notations que dans les preuves précédentes. En particulier, soient σ et σ^* des affectations optimales des clients à X et X^*, respectivement.

La première inégalité se prouve facilement en considérant, pour chaque $y \in X^*$, l'opération consistant en l'ajout de y à X, qui augmente le coût d'au plus $f_y + \sum_{j\in\mathcal{D}:\sigma^*(j)=y}(c_{\sigma^*(j)j} - c_{\sigma(j)j})$. En sommant ces valeurs, on obtient que $c_F(X^*) + c_S(X^*) - c_S(X)$ est non négatif.

Soit de nouveau $\pi : \mathcal{D} \to \mathcal{D}$ une bijection telle que, pour tout $j \in \mathcal{D}$:

- $\sigma^*(\pi(j)) = \sigma^*(j)$;
- si $\sigma(\pi(j)) = \sigma(j)$ alors $\sigma(j)$ capture $\sigma^*(j)$ et $\pi(j) = j$.

Une telle application π peut être obtenue comme dans la preuve du théorème 22.17 en posant $\pi(j) := j$ pour $|\{j \in \mathcal{D} : \sigma^*(j) = y, \sigma(j) = x\}| - |\{j \in \mathcal{D} : \sigma^*(j) = y, \sigma(j) \neq x\}|$ éléments $j \in \mathcal{D}$ tels que $\sigma^*(j) = y$ et $\sigma(j) = x$ pour toute paire $x \in X$, $y \in X^*$ telle que x capture y.

Afin de borner le coût de construction de X, considérons $x \in X$ et définissons $\mathcal{D}_x := \{j \in \mathcal{D} : \sigma(j) = x\}$. Si x ne capture aucun $y \in X^*$, nous abandonnons x et réaffectons chaque $j \in \mathcal{D}_x$ à $\sigma(\pi(j)) \in X \setminus \{x\}$. Alors

$$0 \leq -f_x + \sum_{j \in \mathcal{D}_x} (c_{\sigma(\pi(j))j} - c_{xj}). \tag{22.10}$$

Si l'ensemble $C(\{x\})$ des installations capturées par x est non vide, considérons une plus proche installation $y \in C(\{x\})$ (c-à-d. telle que $\min_{j \in \mathcal{D}}(c_{xj} + c_{yj})$ soit minimum). Nous considérons l'ajout de chaque installation $y' \in C(\{x\}) \setminus \{y\}$, qui augmente le coût d'au moins zéro et d'au plus

$$f_{y'} + \sum_{j \in \mathcal{D}_x : \sigma^*(j) = y', \pi(j) = j} (c_{\sigma^*(j)j} - c_{xj}). \tag{22.11}$$

De plus, nous considérons l'échange $(\{x\}, \{y\})$. Pour $j \in \mathcal{D}_x$, nous réaffectons j à $\sigma(\pi(j))$ si $\pi(j) \neq j$ et sinon à y.

Le nouveau coût de fonctionnement pour $j \in \mathcal{D}_x$ est inférieur ou égal à $c_{\sigma(\pi(j))j}$ dans le premier cas, à $c_{\sigma^*(j)j}$ si $\pi(j) = j$ et $\sigma^*(j) = y$ et sinon à

$$c_{yj} \leq c_{xj} + \min_{k \in \mathcal{D}}(c_{xk} + c_{yk}) \leq c_{xj} + \min_{k \in \mathcal{D}}(c_{xk} + c_{\sigma^*(j)k}) \leq 2c_{xj} + c_{\sigma^*(j)j},$$

où la seconde inégalité est vérifiée, car x capture $\sigma^*(j)$ si $\pi(j) = j$.

En tout, l'échange de x à y augmente le coût d'au moins zéro et d'au plus

$$\begin{aligned} & f_y - f_x - \sum_{j \in \mathcal{D}_x} c_{xj} + \sum_{j \in \mathcal{D}_x : \pi(j) \neq j} c_{\sigma(\pi(j))j} \\ & + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) = y} c_{\sigma^*(j)j} + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) \neq y} (2c_{xj} + c_{\sigma^*(j)j}). \end{aligned} \tag{22.12}$$

En ajoutant (22.11) et (22.12), qui sont tous deux non négatifs, on obtient

$$\begin{aligned} 0 \quad \leq \quad & \sum_{y' \in C(\{x\})} f_{y'} - f_x + \sum_{j \in \mathcal{D}_x : \pi(j) \neq j} (c_{\sigma(\pi(j))j} - c_{xj}) \\ & + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) = y} (c_{\sigma^*(j)j} - c_{xj}) + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) \neq y} 2c_{\sigma^*(j)j} \\ \leq \quad & \sum_{y' \in C(\{x\})} f_{y'} - f_x + \sum_{j \in \mathcal{D}_x : \pi(j) \neq j} (c_{\sigma(\pi(j))j} - c_{xj}) + 2 \sum_{j \in \mathcal{D}_x : \pi(j) = j} c_{\sigma^*(j)j}. \end{aligned} \tag{22.13}$$

En sommant (22.10) et (22.13), respectivement, sur tous les $x \in X$, on obtient

$$\begin{aligned} 0 \quad \leq \quad & \sum_{x \in X} \sum_{y' \in C(\{x\})} f_{y'} - c_F(X) + \sum_{j \in \mathcal{D} : \pi(j) \neq j} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\ & + 2 \sum_{j \in \mathcal{D} : \pi(j) = j} c_{\sigma^*(j)j} \\ \leq \quad & c_F(X^*) - c_F(X) + \sum_{j \in \mathcal{D} : \pi(j) \neq j} (c_{\sigma^*(j)j} + c_{\sigma^*(j)\pi(j)} + c_{\sigma(\pi(j))\pi(j)} - c_{\sigma(j)j}) \\ & + 2 \sum_{j \in \mathcal{D} : \pi(j) = j} c_{\sigma^*(j)j} \\ = \quad & c_F(X^*) - c_F(X) + 2c_S(X^*). \end{aligned}$$

□

D'après le lemme 22.15, cela implique un algorithme d'approximation de facteur $(3 + \epsilon)$ pour tout $\epsilon > 0$. En combinant cela avec le théorème 22.10, nous obtenons un algorithme d'approximation de facteur $2,375$ (exercice 12). Charikar et Guha [2005] ont prouvé la même garantie de performance pour un algorithme de recherche locale semblable.

22.7 Problèmes de localisation avec capacités

Un des principaux avantages des algorithmes de recherche locale est leur flexibilité. Ils peuvent être appliqués avec des fonctions coût arbitraires et même en introduisant des contraintes supplémentaires complexes. Pour les problèmes de localisation avec capacités strictes, la recherche locale est la seule technique actuellement connue qui fournisse une garantie de performance.

Il y a plusieurs problèmes de localisation avec capacités. Mahdian et Pál [2003] ont défini le problème général suivant, qui contient plusieurs cas particuliers importants :

PROBLÈME DE LOCALISATION UNIVERSEL

Instance Un ensemble fini $\mathcal{D}$ de clients et un ensemble fini $\mathcal{F}$ d'installations potentielles ; une métrique c sur $V := \mathcal{D} \cup \mathcal{F}$, c.-à-d. des distances $c_{ij} \geq 0$ $(i, j \in V)$ telles que $c_{ii} = 0$, $c_{ij} = c_{ji}$ et $c_{ij} + c_{jk} \geq c_{ik}$ pour tout $i, j, k \in V$; une demande $d_j \geq 0$ pour chaque $j \in \mathcal{D}$ et pour chaque $i \in \mathcal{F}$, une fonction coût $f_i : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{\infty\}$ continue à gauche et non décroissante.

Tâche Trouver $x_{ij} \in \mathbb{R}_+$ pour $i \in \mathcal{F}$ et $j \in \mathcal{D}$, tel que $\sum_{i \in \mathcal{F}} x_{ij} = d_j$ pour tout $j \in \mathcal{D}$ et que $c(x) := c_F(x) + c_S(x)$ soit minimum, où

$$c_F(x) := \sum_{i \in \mathcal{F}} f_i\left(\sum_{j \in \mathcal{D}} x_{ij}\right) \qquad \text{et} \qquad c_S(x) := \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}.$$

$f_i(z)$ peut s'interpréter comme le coût de mise en place de la capacité z pour l'installation i. Nous devons spécifier comment les fonctions f_i sont données. Nous supposons l'existence d'un oracle qui, pour chaque $i \in \mathcal{F}$, $u, c \in \mathbb{R}_+$ et $t \in \mathbb{R}$, calcule $f_i(u)$ et $\max\{\delta \in \mathbb{R} : u + \delta \geq 0,\ f_i(u + \delta) - f_i(u) + c|\delta| \leq t\}$. C'est une hypothèse naturelle puisqu'un oracle peut être aisément implémenté pour les cas particuliers les plus importants du PROBLÈME DE LOCALISATION UNIVERSEL. Ces cas particuliers sont :

- le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS. Dans ce cas, $d_j = 1$ $(j \in \mathcal{D})$, $f_i(0) = 0$ et $f_i(z) = t_i$ pour un certain $t_i \in \mathbb{R}_+$ et pour tout $z > 0$ $(i \in \mathcal{F})$;

- le PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS. Dans ce cas, $f_i(0) = 0$, $f_i(z) = t_i$ pour $0 < z \leq u_i$ et $f_i(z) = \infty$ pour $z > u_i$, où $u_i, t_i \in \mathbb{R}_+$ $(i \in \mathcal{F})$;
- le PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS SOUPLES. Dans ce cas, $d_j = 1$ $(j \in \mathcal{D})$ et il existe $u_i \in \mathbb{N}$ et $t_i \in \mathbb{R}_+$ tels que $f_i(z) = \lceil \frac{z}{u_i} \rceil t_i$ pour tout $z \geq 0$ $(i \in \mathcal{F})$.

Remarquons que, dans le premier et le troisième cas, il existe toujours une solution optimale entière. Cela est évident pour le premier cas. On le voit facilement dans le troisième cas en prenant une solution optimale arbitraire y et en appliquant le résultat suivant avec $d_j = 1$ pour $j \in \mathcal{D}$ et $z_i = \max\{z : f_i(z) \leq f_i(\sum_{j \in \mathcal{D}} y_{ij})\} \in \mathbb{Z}_+$ pour $i \in \mathcal{F}$:

Proposition 22.21. *Soient $\mathcal{D}$ et $\mathcal{F}$ des ensembles finis, $d_j \geq 0$ $(j \in \mathcal{D})$, $z_i \geq 0$ $(i \in \mathcal{F})$ et $c_{ij} \geq 0$ $(i \in \mathcal{F}, j \in \mathcal{D})$ tels que $\sum_{j \in \mathcal{D}} d_j \leq \sum_{i \in \mathcal{F}} z_i$. Alors une solution optimale de*

$$\min \left\{ \sum_{i \in \mathcal{F},\, j \in \mathcal{D}} c_{ij} x_{ij} : x \geq 0, \sum_{i \in \mathcal{F}} x_{ij} = d_j \ (j \in \mathcal{D}), \sum_{j \in \mathcal{D}} x_{ij} \leq z_i \ (i \in \mathcal{F}) \right\} \tag{22.14}$$

peut être trouvée en temps $O(n^3 \log n)$, où $n = |\mathcal{D}| + |\mathcal{F}|$. Si tous les d_j et z_i sont des entiers, alors il existe une solution optimale entière.

Preuve. (22.14) est équivalent à l'instance (G, b, c) du PROBLÈME DE HITCHCOCK, définie par $G := (A \,\dot\cup\, B, A \times B)$, $A := \{v_j : j \in \mathcal{D}\} \,\dot\cup\, \{0\}$, $B := \{w_i : i \in \mathcal{F}\}$, $b(v_j) := d_j$ pour $j \in \mathcal{D}$, $b(w_i) = -z_i$ pour $i \in \mathcal{F}$, $b(0) := \sum_{i \in \mathcal{F}} z_i - \sum_{j \in \mathcal{D}} d_j$, $c(v_j, w_i) := c_{ij}$ et $c(0, w_i) := 0$ pour $i \in \mathcal{F}$ et $j \in \mathcal{D}$. Ainsi (22.14) peut être résolu en temps $O(n^3 \log n)$ d'après le théorème 9.16. Si b est entier, l'ALGORITHME PAR ÉLIMINATION DU CYCLE MOYEN MINIMUM et l'ALGORITHME PAR PLUS COURTS CHEMINS SUCCESSIFS calculent des solutions optimales entières. □

La version avec des capacités souples peut être réduite assez facilement à la version sans capacités, à l'aide d'une technique qui a été proposée à l'origine par Jain et Vazirani [2001] :

Théorème 22.22. (Mahdian, Ye et Zhang [2006]) *Soient γ_F et γ_S des constantes et A un algorithme polynomial tels que, pour toute instance du* PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS, *A calcule une solution X telle que $c_F(X) + c_S(X) \leq \gamma_F c_F(X^*) + \gamma_S c_S(X^*)$ pour tout $\emptyset \neq X^* \subseteq \mathcal{F}$. Alors il existe un algorithme d'approximation de facteur $(\gamma_F + \gamma_S)$ pour le* PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS SOUPLES.

Preuve. Considérons une instance $I = (\mathcal{F}, \mathcal{D}, (c_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}}, (f_i)_{i \in \mathcal{F}})$ du PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS SOUPLES, avec l'hypothèse $f_i(z) = \lceil \frac{z}{u_i} \rceil t_i$ pour $i \in \mathcal{F}$ et $z \in \mathbb{R}_+$. Nous allons la transformer en

une instance $I' = (\mathcal{F}, \mathcal{D}, (f'_i)_{i\in\mathcal{F}}, (c'_{ij})_{i\in\mathcal{F},j\in\mathcal{D}})$ du PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS en posant $f'_i := t_i$ et $c'_{ij} := c_{ij} + \frac{t_i}{u_i}$ pour $i \in \mathcal{F}$ et $j \in \mathcal{D}$. (Remarquons que c' est métrique dès que c est métrique.)

Nous appliquons A à I' et trouvons une solution $X \in \mathcal{F}$ et une affectation $\sigma : \mathcal{D} \to X$. Posons $x_{ij} := 1$ si $\sigma(j) = i$ et sinon $x_{ij} := 0$. Si $\sigma^* : \mathcal{D} \to \mathcal{F}$ est une solution optimale de I et $X^* := \{i \in \mathcal{F} : \exists j \in \mathcal{D} : \sigma^*(j) = i\}$ est l'ensemble des installations ouvertes au moins une fois,

$$\begin{aligned}
c_F(x) + c_S(x) &= \sum_{i\in X} \left\lceil \frac{|\{j \in \mathcal{D} : \sigma(j) = i\}|}{u_i} \right\rceil t_i + \sum_{j\in\mathcal{D}} c_{\sigma(j)j} \\
&\leq \sum_{i\in X} t_i + \sum_{j\in\mathcal{D}} c'_{\sigma(j)j} \\
&\leq \gamma_F \sum_{i\in X^*} t_i + \gamma_S \sum_{j\in\mathcal{D}} c'_{\sigma^*(j)j} \\
&\leq (\gamma_F + \gamma_S) \sum_{i\in X^*} \left\lceil \frac{|\{j \in \mathcal{D} : \sigma^*(j) = i\}|}{u_i} \right\rceil t_i + \gamma_S \sum_{j\in\mathcal{D}} c_{\sigma^*(j)j}.
\end{aligned}$$

□

Corollaire 22.23. *Le* PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS SOUPLES *a un algorithme d'approximation de facteur 2,89.*

Preuve. Appliquons le théorème 22.22 à l'ALGORITHME PAR AJUSTEMENT DUAL (corollaire 22.7(c)) ; ici $\gamma_F = 1,11$ et $\gamma_S = 1,78$. □

Voir l'exercice 11 pour un meilleur rapport de performance.

Lorsque nous considérons des capacités strictes, nous devons permettre que les demandes des clients soient divisées, c.-à-d. affectées à plusieurs installations ouvertes : si nous ne permettons pas cela , nous ne pouvons pas espérer le moindre résultat puisque même décider s'il existe une solution réalisable est un problème *NP*-complet (ce problème contient le PROBLÈME DE LA PARTITION ; voir corollaire 15.28).

Le premier algorithme d'approximation pour le PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS est dû à Pál, Tardos et Wexler [2001], qui ont généralisé un résultat obtenu auparavant pour un cas particulier par Korupolu, Plaxton et Rajaraman [2000]. La garantie de performance a été ensuite améliorée jusqu'à 5,83 par Zhang, Chen et Ye [2004]. Dans le cas particulier de coûts de construction uniformes, Levi, Shmoys et Swamy [2004] ont obtenu un algorithme d'approximation de facteur 5 en arrondissant une relaxation linéaire.

Le travail de Pál, Tardos et Wexler [2001] a été généralisé au PROBLÈME DE LOCALISATION UNIVERSEL par Mahdian et Pál [2003]. Ils ont obtenu un algorithme d'approximation de facteur $7,88$. Au paragraphe suivant, nous présentons un algorithme de recherche locale qui permet d'obtenir une garantie de performance égale

à $6,702$ pour le PROBLÈME DE LOCALISATION UNIVERSEL. Mais faisons d'abord la remarque suivante.

Lemme 22.24. (Mahdian et Pál [2003]) *Toute instance du* PROBLÈME DE LOCALISATION UNIVERSEL *a une solution optimale.*

Preuve. S'il n'existe pas de solution de coût fini, alors toute solution est optimale. Sinon considérons $(x^i)_{i\in\mathbb{N}}$ une suite de solutions dont les coûts approchent l'infimum $c^* \in \mathbb{R}_+$ de l'ensemble des coûts des solutions réalisables. Comme cette suite est bornée, il existe une sous-suite $(x^{i_j})_{j\in\mathbb{N}}$ convergeant vers une solution x^*. x^* est réalisable. Comme toutes les fonctions f_i sont continues à gauche et non décroissantes, nous avons $c(x^*) = c(\lim_{j\to\infty} x^{i_j}) \leq \lim_{j\to\infty} c(x^{i_j}) = c^*$, c.-à-d. x^* est optimale. □

22.8 Problème de localisation universel

Dans ce paragraphe, fondé sur l'article de Vygen [2007], nous présentons un algorithme de recherche locale pour le PROBLÈME DE LOCALISATION UNIVERSEL. Il utilise deux opérations. Tout d'abord, pour $t \in \mathcal{F}$ et $\delta \in \mathbb{R}_+$, nous considérons l'opération ADD(t,δ), qui consiste à remplacer la solution réalisable courante x par une solution optimale y du problème suivant :

$$\min\Bigg\{ c_S(y) \;:\; y_{ij} \geq 0 \,(i\in\mathcal{F},\, j\in\mathcal{D}),\ \sum_{i\in\mathcal{F}} y_{ij} = d_j \,(j\in\mathcal{D}),\\ \sum_{j\in\mathcal{D}} y_{ij} \leq \sum_{j\in\mathcal{D}} x_{ij} \,(i\in\mathcal{F}\setminus\{t\}),\ \sum_{j\in\mathcal{D}} y_{tj} \leq \sum_{j\in\mathcal{D}} x_{tj} + \delta \Bigg\}. \tag{22.15}$$

Nous notons $c^x(t,\delta) := c_S(y) - c_S(x) + f_t(\sum_{j\in\mathcal{D}} x_{tj} + \delta) - f_t(\sum_{j\in\mathcal{D}} x_{tj})$ le coût estimé de cette opération. C'est une borne supérieure de $c(y) - c(x)$.

Lemme 22.25. (Mahdian et Pál [2003]) *Soit $\epsilon > 0$. Soit x une solution réalisable d'une instance donnée et soit $t \in \mathcal{F}$. Alors il existe un algorithme, dont le temps de calcul est $O(|V|^3 \log|V| \epsilon^{-1})$, qui trouve un $\delta \in \mathbb{R}_+$ tel que $c^x(t,\delta) \leq -\epsilon c(x)$ ou conclut qu'il n'existe aucun $\delta \in \mathbb{R}_+$ tel que $c^x(t,\delta) \leq -2\epsilon c(x)$.*

Preuve. Nous pouvons supposer que $c(x) > 0$. Considérons $C := \{\nu\epsilon c(x) : \nu \in \mathbb{Z}_+, \nu \leq \lceil \frac{1}{\epsilon} \rceil\}$. Pour chaque $\gamma \in C$, soit δ_γ le $\delta \in \mathbb{R}_+$ maximum tel que $f_t(\sum_{j\in\mathcal{D}} x_{tj} + \delta) - f_t(\sum_{j\in\mathcal{D}} x_{tj}) \leq \gamma$. Nous calculons $c^x(t,\delta_\gamma)$ pour tout $\gamma \in C$.

Supposons qu'il existe un $\delta \in \mathbb{R}_+$ tel que $c^x(t,\delta) \leq -2\epsilon c(x)$. Alors considérons

$$\gamma := \epsilon c(x) \left\lceil \frac{1}{\epsilon c(x)} \left(f_t\left(\sum_{j\in\mathcal{D}} x_{tj} + \delta \right) - f_t\left(\sum_{j\in\mathcal{D}} x_{tj} \right) \right) \right\rceil \in C.$$

Remarquons que $\delta_\gamma \geq \delta$ et ainsi $c^x(t,\delta_\gamma) < c^x(t,\delta) + \epsilon c(x) \leq -\epsilon c(x)$.

Le temps de calcul est dominé par la résolution de $|C|$ problèmes du type (22.15). Alors le temps de calcul est une conséquence de la proposition 22.21. □

S'il n'existe pas d'opération ADD suffisamment profitable, le coût de fonctionnement peut être borné. Le résultat suivant est principalement dû à Pál, Tardos et Wexler [2001] :

Lemme 22.26. *Soit $\epsilon > 0$ et soient x, x^* des solutions réalisables d'une instance donnée et supposons $c^x(t,\delta) \geq -\frac{\epsilon}{|\mathcal{F}|}c(x)$ pour tout $t \in \mathcal{F}$ et $\delta \in \mathbb{R}_+$. Alors $c_S(x) \leq c_F(x^*) + c_S(x^*) + \epsilon c(x)$.*

Preuve. Considérons le graphe orienté (biparti complet) $G = (\mathcal{D} \dot\cup \mathcal{F}, (\mathcal{D}\times\mathcal{F}) \cup (\mathcal{F}\times\mathcal{D}))$ avec des poids associés aux arcs $c((j,i)) := c_{ij}$ et $c((i,j)) := -c_{ij}$ pour $i \in \mathcal{F}$ et $j \in \mathcal{D}$. Définissons $b(i) := \sum_{j\in\mathcal{D}}(x_{ij} - x^*_{ij})$ pour $i \in \mathcal{F}$, $S := \{i \in \mathcal{F} : b(i) > 0\}$ et $T := \{i \in \mathcal{F} : b(i) < 0\}$.

Définissons un b-flot $g : E(G) \to \mathbb{R}_+$ par $g(i,j) := \max\{0, x_{ij} - x^*_{ij}\}$ et $g(j,i) := \max\{0, x^*_{ij} - x_{ij}\}$ pour $i \in \mathcal{F}$, $j \in \mathcal{D}$.

Écrivons g comme la somme de b_t-flots g_t pour $t \in T$, tels que $b_t(t) = b(t)$, $b_t(v) = 0$ pour $v \in T \setminus \{t\}$ et $0 \leq b_t(v) \leq b(v)$ pour $v \in V(G) \setminus T$. (Cela peut être réalisé par des techniques de décomposition de flot.)

Pour chaque $t \in T$, g_t définit une manière possible de réaffecter les clients à t, c.-à-d. une nouvelle solution x^t définie par $x^t_{ij} := x_{ij} + g_t(j,i) - g_t(i,j)$ pour $i \in \mathcal{F}$, $j \in \mathcal{D}$. Nous avons $c_S(x^t) = c_S(x) + \sum_{e\in E(G)} c(e)g_t(e)$ et ainsi

$$c^x(t, -b(t)) \leq \sum_{e\in E(G)} c(e)g_t(e) + f_t\left(\sum_{j\in\mathcal{D}} x^*_{tj}\right) - f_t\left(\sum_{j\in\mathcal{D}} x_{tj}\right).$$

Si le terme de gauche est supérieur ou égal à $-\frac{\epsilon}{|\mathcal{F}|}c(x)$ pour tout $t \in T$, on obtient en sommant

$$\begin{aligned} -\epsilon c(x) &\leq \sum_{e\in E(G)} c(e)g(e) + \sum_{t\in T} f_t\left(\sum_{j\in\mathcal{D}} x^*_{tj}\right) \\ &\leq \sum_{e\in E(G)} c(e)g(e) + c_F(x^*) \\ &= c_S(x^*) - c_S(x) + c_F(x^*). \end{aligned}$$

□

Nous allons maintenant décrire le second type d'opération. Soit x une solution réalisable pour une instance donnée du PROBLÈME DE LOCALISATION UNIVERSEL. Soit A une arborescence telle que $V(A) \subseteq \mathcal{F}$ et $\delta \in \Delta^x_A := \{\delta \in \mathbb{R}^{V(A)} : \sum_{j\in\mathcal{D}} x_{ij} + \delta_i \geq 0$ pour tout $i \in V(A)$, $\sum_{i\in V(A)} \delta_i = 0\}$.

Alors nous considérons l'opération PIVOT(A,δ), qui consiste à remplacer x par une solution x' telle que $\sum_{j\in\mathcal{D}} x'_{ij} = \sum_{j\in\mathcal{D}} x_{ij} + \delta_i$ pour $i \in V(A)$,

$\sum_{j\in\mathcal{D}} x'_{ij} = \sum_{j\in\mathcal{D}} x_{ij}$ pour $i \in \mathcal{F} \setminus V(A)$ et $c(x') \leq c(x) + c^x(A,\delta)$, où $c^x(A,\delta) := \sum_{i\in V(A)} c^x_{A,i}(\delta)$ et

$$c^x_{A,i}(\delta) := f_i\left(\sum_{j\in\mathcal{D}} x_{ij} + \delta_i\right) - f_i\left(\sum_{j\in\mathcal{D}} x_{ij}\right) + \left|\sum_{l\in A_i^+} \delta_l\right| c_{ip(i)}$$

pour $i \in V(A)$. On note ici A_i^+ l'ensemble des sommets que l'on peut atteindre depuis i dans A et on note $p(i)$ le prédécesseur de i dans A (et arbitrairement si i est la racine). Une telle solution x' peut être construite aisément en déplaçant les demandes le long des arcs de A dans l'ordre topologique inverse. Remarquons que l'orientation de A n'a pas de signification à l'égard de $c^x(A,\delta)$ et est seulement utilisée pour simplifier les notations.

L'opération ne sera effectuée que si son *coût estimatif* $c^x(A,\delta)$ est suffisamment négatif. Cela garantit que l'algorithme de recherche locale s'arrête après un nombre polynomial d'étapes d'amélioration. Le *coût estimatif de routage* de PIVOT(A,δ) correspond à $\sum_{i\in V(A)} \left|\sum_{l\in A_i^+} \delta_l\right| c_{ip(i)}$.

Nous allons maintenant montrer comment trouver une opération PIVOT qui améliore la solution à moins que nous soyons à un optimum local :

Lemme 22.27. (Vygen [2007]) *Soit $\epsilon > 0$ et soit A une arborescence telle que $V(A) \subseteq \mathcal{F}$. Soit x une solution réalisable. Alors il existe un algorithme, dont le temps de calcul est $O(|\mathcal{F}|^4\epsilon^{-3})$, qui trouve un $\delta \in \Delta_A^x$ tel que $c^x(A,\delta) \leq -\epsilon c(x)$ ou conclut qu'il n'existe aucun $\delta \in \Delta_A^x$ tel que $c^x(A,\delta) \leq -2\epsilon c(x)$.*

Preuve. Numérotons $V(A) = \{1,\dots,n\}$ dans l'ordre topologique inverse ; donc pour tout $(i,j) \in E(A)$, nous avons $i > j$. Pour $k \in V(A)$ tel que $(p(k),k) \in E(A)$, considérons $B(k) := \{i < k : (p(k),i) \in E(A)\}$ l'ensemble des «petits frères» de k et posons $B(k) := \emptyset$ si k est la racine de A. Définissons $I_k := \bigcup_{l\in B(k)\cup\{k\}} A_l^+$, $b(k) := \max(\{0\} \cup B(k))$ et $s(k) := \max(\{0\} \cup (A_k^+ \setminus \{k\}))$.

Considérons $C := \{\nu\frac{\epsilon}{n}c(x) : \nu \in \mathbb{Z}, -\lceil\frac{n}{\epsilon}\rceil - n \leq \nu \leq \lceil\frac{n}{\epsilon}\rceil + n\}$. Nous calculons la table $(T_A^x(k,\gamma))_{k\in\{0,\dots,n\},\gamma\in C}$, définie de la manière suivante. Posons $T_A^x(0,0) := 0$, $T_A^x(0,\gamma) := \emptyset$ pour tout $\gamma \in C \setminus \{0\}$ et pour $k = 1,\dots,n$, nous définissons $T_A^x(k,\gamma)$ par une solution optimale $\delta \in \mathbb{R}^{I_k}$ de

$$\max\Bigg\{\sum_{i\in I_k}\delta_i : \gamma' \in C,\ T_A^x(b(k),\gamma') \neq \emptyset,\ \delta_i = (T_A^x(b(k),\gamma'))_i \text{ pour } i \in \bigcup_{l\in B(k)} A_l^+,$$
$$\gamma'' \in C,\ T_A^x(s(k),\gamma'') \neq \emptyset,\ \delta_i = (T_A^x(s(k),\gamma''))_i \text{ pour } i \in A_k^+ \setminus \{k\},$$
$$\sum_{j\in\mathcal{D}} x_{kj} + \delta_k \geq 0,\ \gamma' + \gamma'' + c^x_{A,k}(\delta) \leq \gamma\Bigg\}$$

si l'ensemble sur lequel on prend le maximum est non vide et sinon $T_A^x(k,\gamma) := \emptyset$.

$-\sum_{i\in I_k}(T_A^x(k,\gamma))_i$ est l'excès minimal obtenu au niveau du prédécesseur $p(k)$ de k lorsque l'on déplace la demande de chaque sommet de I_k à son prédécesseur respectif ou vice versa, à un coût estimatif total inférieur ou égal à γ.

Remarquons que $T_A^x(k,0) \neq \emptyset$ pour $k = 0, \ldots, n$. Ainsi, nous pouvons choisir le minimum $\gamma \in C$ tel que $T_A^x(n,\gamma) \neq \emptyset$ et $\sum_{i=1}^n (T_A^x(n,\gamma))_i \geq 0$. Nous choisissons alors $\delta \in \Delta_A^x$ tel que $\delta_i = (T_A^x(n,\gamma))_i$ ou $0 \leq \delta_i \leq (T_A^x(n,\gamma))_i$ pour tout $i = 1, \ldots, n$ et $|\sum_{l \in A_i^+} \delta_l| \leq |\sum_{l \in A_i^+} (T_A^x(n,\gamma))_l|$ pour tout $i = 1, \ldots, n$. Cela peut être fait en posant $\delta := T_A^x(n,\gamma)$ et en diminuant successivement δ_i pour le i maximum tel que $\delta_i > 0$ et tel que $\sum_{l \in A_k^+} \delta_l > 0$ pour tous les sommets k du chemin allant de n à i dans A. Remarquons que la propriété $c^x(A,\delta) \leq \gamma$ est conservée. Il reste à montrer que γ est suffisamment petit.

Supposons qu'il existe une opération PIVOT(A,δ) telle que $c^x(A,\delta) \leq -2\epsilon c(x)$. Comme $c_{A,i}^x(\delta) \geq -f_i(\sum_{j \in \mathcal{D}} x_{ij}) \geq -c(x)$ pour tout $i \in V(A)$, cela implique également $c_{A,i}^x(\delta) < c_F(x) \leq c(x)$. Ainsi $\gamma_i := \left\lceil c_{A,i}^x(\delta) \frac{n}{\epsilon c(x)} \right\rceil \frac{\epsilon c(x)}{n} \in C$ pour $i = 1, \ldots, n$ et $\sum_{i \in I} \gamma_i \in C$ pour tout $I \subseteq \{1, \ldots, n\}$. Alors on peut montrer facilement par induction que $\sum_{i \in I_k} (T_A^x(k, \sum_{l \in I_k} \gamma_l))_i \geq \sum_{i \in I_k} \delta_i$ pour $k = 1, \ldots, n$. Ainsi on trouve une opération PIVOT de coût estimatif inférieur ou égal à $\sum_{i=1}^n \gamma_i < c^x(A,\delta) + \epsilon c(x) \leq -\epsilon c(x)$.

Le temps de calcul peut être estimé de la manière suivante. Nous devons calculer les $n|C|$ entrées de la table et pour chaque entrée $T_A^x(k,\gamma)$ nous essayons toutes les valeurs de $\gamma', \gamma'' \in C$. Cela fournit les valeurs δ_i pour $i \in I_k \setminus \{k\}$. L'étape principale est le calcul du δ_k maximum pour lequel $\gamma' + \gamma'' + c_{A,k}^x(\delta) \leq \gamma$. Cela peut être fait directement à l'aide de l'oracle dont nous avons supposé l'existence pour les fonctions f_i, $i \in \mathcal{F}$. Le calcul final de δ à partir de $T_A^x(n,\gamma)$, $\gamma \in C$, s'effectue facilement en temps linéaire. Ainsi le temps de calcul total est $O(n|C|^3) = O(|\mathcal{F}|^4 \epsilon^{-3})$. □

Nous allons considérer l'opération PIVOT(A,δ) pour des arborescences particulières : les étoiles et les comètes. A sera appelée l'**étoile de centre** v si $A = (\mathcal{F}, \{(v,w) : w \in \mathcal{F} \setminus \{v\}\})$ et sera appelée la **comète de centre** v **et de queue** (t,s) si $A = (\mathcal{F}, \{(t,s)\} \cup \{(v,w) : w \in \mathcal{F} \setminus (v,s)\})$ et si v, t, s sont des éléments de $\mathcal{F}$ distincts. Remarquons qu'il existe au plus $|\mathcal{F}|^3$ étoiles et comètes.

Nous allons maintenant montrer qu'un optimum local (approché) a un coût de construction assez faible.

Lemme 22.28. *Soient x, x^* des solutions réalisables d'une instance donnée et supposons $c^x(A,\delta) \geq -\frac{\epsilon}{|\mathcal{F}|} c(x)$ pour toutes les étoiles et comètes A et $\delta \in \Delta_A^x$. Alors $c_F(x) \leq 4c_F(x^*) + 2c_S(x^*) + 2c_S(x) + \epsilon c(x)$.*

Preuve. Nous utilisons les notations du lemme 22.26 et considérons l'instance suivante du PROBLÈME DE HITCHCOCK :

$$\begin{array}{lll} \min & \displaystyle\sum_{s \in S, t \in T} c_{st} y(s,t) & \\ \text{s.c.} & \displaystyle\sum_{t \in T} y(s,t) = b(s) & (s \in S) \\ & \displaystyle\sum_{s \in S} y(s,t) = -b(t) & (t \in T) \\ & y(s,t) \geq 0 & (s \in S, t \in T). \end{array} \tag{22.16}$$

D'après la proposition 9.19 il existe une solution optimale $y : S \times T \to \mathbb{R}_+$ de (22.16) telle que $F := (S \cup T, \{\{s,t\} : y(s,t) > 0\})$ soit une forêt.

Comme $(b_t(s))_{s\in S, t\in T}$ est une solution réalisable de (22.16), nous avons

$$\begin{aligned}\sum_{s\in S,t\in T} c_{st}y(s,t) &\leq \sum_{s\in S,t\in T} c_{st}b_t(s)\\ &= \sum_{s\in S,t\in T} c_{st}(g_t(\delta^+(s)) - g_t(\delta^-(s)))\\ &\leq \sum_{e\in E(G)} |c(e)|g(e)\\ &\leq c_S(x^*) + c_S(x).\end{aligned} \tag{22.17}$$

Nous allons maintenant définir au plus $|\mathcal{F}|$ opérations PIVOT. Nous dirons qu'une opération PIVOT(A,δ) *ferme* $s \in S$ (resp. à x et x^*) si $\sum_{j\in\mathcal{D}} x_{sj} > \sum_{j\in\mathcal{D}} x_{sj} + \delta_s = \sum_{j\in\mathcal{D}} x^*_{sj}$. Nous dirons qu'elle *ouvre* $t \in T$ si $\sum_{j\in\mathcal{D}} x_{tj} < \sum_{j\in\mathcal{D}} x_{tj} + \delta_t \leq \sum_{j\in\mathcal{D}} x^*_{tj}$. Au cours de toutes les opérations que nous allons définir, chaque $s \in S$ sera fermé une fois et chaque $t \in T$ sera ouvert au plus quatre fois. De plus, le coût estimatif total du routage sera inférieur ou égal à $2\sum_{s\in S,t\in T} c_{st}y(s,t)$. Ainsi, le coût estimatif total des opérations sera inférieur ou égal à $4c_F(x^*) + 2c_S(x^*) + 2c_S(x) - c_F(x)$. Cela prouvera le lemme.

Pour définir les opérations, orientons F comme une ramification B dont chaque composante a pour racine un élément de T. Notons $y(e) := y(s,t)$ si $e \in E(B)$ a pour extrémités $s \in S$ et $t \in T$. Un sommet $v \in V(B)$ est dit *faible* si $y(\delta^+_B(v)) > y(\delta^-_B(v))$ et sinon *fort*. Nous noterons par $\Gamma^+(v)$ l'ensemble des enfants de $v \in V(B)$ dans B et par respectivement $\Gamma^+_s(v)$ et $\Gamma^+_w(v)$ l'ensemble des enfants forts et faibles de $v \in V(B)$ dans B.

Soit $t \in T$ et soit $\Gamma^+_w(t) = \{w_1, \ldots, w_k\}$ l'ensemble des enfants faibles de t ordonné de telle manière que $r(w_1) \leq \cdots \leq r(w_k)$, où $r(w_i)$ est défini par : $r(w_i) := \max\Big\{0, y(w_i,t) - \sum_{t'\in\Gamma^+_w(w_i)} y(w_i,t')\Big\}$. De plus, nous ordonnons $\Gamma^+_s(t) = \{s_1, \ldots, s_l\}$ de telle manière que $y(s_1,t) \geq \cdots \geq y(s_l,t)$.

Supposons d'abord $k > 0$. Pour $i = 1, \ldots, k-1$, considérons une opération PIVOT avec l'étoile de centre w_i, acheminant

- au plus $2y(w_i,t')$ unités de demande de w_i jusqu'à chaque enfant faible t' de w_i,
- $y(w_i,t')$ unités de w_i jusqu'à chaque enfant fort t' de w_i et
- $r(w_i)$ unités de w_i à $\Gamma^+_s(w_{i+1})$,

fermant w_i et ouvrant un sous-ensemble de $\Gamma^+(w_i) \cup \Gamma^+_s(w_{i+1})$. Le coût estimatif du routage est inférieur ou égal à

$$\begin{aligned}&\sum_{t'\in\Gamma^+_w(w_i)} c_{w_it'}2y(w_i,t') + \sum_{t'\in\Gamma^+_s(w_i)} c_{w_it'}y(w_i,t') + c_{tw_i}r(w_i)\\ &\quad + c_{tw_{i+1}}r(w_{i+1}) + \sum_{t'\in\Gamma^+_s(w_{i+1})} c_{w_{i+1}t'}y(w_{i+1},t'),\end{aligned}$$

puisque $r(w_i) \leq r(w_{i+1}) \leq \sum_{t' \in \Gamma_s^+(w_{i+1})} y(w_{i+1}, t')$.

Afin de définir davantage d'opérations PIVOT liées à t, nous distinguons trois cas.

Cas 1 : t est fort ou $l = 0$. Alors considérons :

(1) une opération PIVOT avec l'étoile de centre w_k, acheminant
- $y(w_k, t')$ unités de demande de w_k jusqu'à chaque enfant t' de w_k et
- $y(w_k, t)$ unités de w_k à t,

fermant w_k et ouvrant t et les enfants de w_k et

(2) une opération PIVOT avec l'étoile de centre t, acheminant
- au plus $2y(s, t)$ unités de chaque enfant fort s de t à t,

fermant les enfants forts de t et ouvrant t.

(Dans le cas $l = 0$, le second PIVOT peut être omis.)

Cas 2 : t est faible, $l \geq 1$ et $y(w_k, t) + y(s_1, t) \geq \sum_{i=2}^{l} y(s_i, t)$. Alors considérons :

(1) une opération PIVOT avec l'étoile de centre w_k, acheminant
- $y(w_k, t')$ unités de demande de w_k jusqu'à chaque enfant t' de w_k et
- $y(w_k, t)$ unités de w_k à t,

fermant w_k, ouvrant les enfants de w_k et ouvrant t,

(2) une opération PIVOT avec l'étoile de centre s_1, acheminant
- $y(s_1, t')$ unités de s_1 jusqu'à chaque enfant t' de s_1 et
- $y(s_1, t)$ unités de s_1 à t,

fermant s_1, ouvrant les enfants de s_1 et ouvrant t et

(3) une opération PIVOT avec l'étoile de centre t, acheminant
- au plus $2y(s_i, t)$ unités de s_i à t pour $i = 2, \ldots, l$,

fermant $s_2, \ldots, s_l$ et ouvrant t.

Cas 3 : t est faible, $l \geq 1$ et $y(w_k, t) + y(s_1, t) < \sum_{i=2}^{l} y(s_i, t)$. Alors considérons :

(1) une opération PIVOT avec la comète de centre w_k et de queue (t, s_1), acheminant
- $y(w_k, t')$ unités de demande de w_k jusqu'à chaque enfant t' de w_k,
- $y(w_k, t)$ unités de w_k à t et
- au plus $2y(s_1, t)$ unités de s_1 à t,

fermant w_k et s_1 et ouvrant t et les enfants de w_k,

(2) une opération PIVOT avec l'étoile de centre t, acheminant
- au plus $2y(s_i, t)$ unités de s_i à t pour chaque élément impair i de $\{2, \ldots, l\}$,

fermant les éléments impairs de $\{s_2, \ldots, s_l\}$ et ouvrant t et

(3) une opération PIVOT avec l'étoile de centre t, acheminant
- au plus $2y(s_i, t)$ unités de s_i à t pour chaque élément pair i de $\{2, \ldots, l\}$,

fermant les éléments pairs de $\{s_2, \ldots, s_l\}$ et ouvrant t.

Dans le cas où $k = 0$, nous considérons les mêmes opérations PIVOT, sauf que nous omettons l'opération (1) aux cas 1 et 2 (où $y(w_0, t) := 0$) et la remplaçons par une opération PIVOT avec l'étoile de centre t du cas 3, acheminant au plus $2y(s_1, t)$ unités de s_1 à t, fermant s_1 et ouvrant t.

Nous collectons toutes ces opérations PIVOT pour tout $t \in T$. Alors, en tout, nous avons fermé chaque $s \in S$ une fois et ouvert chaque $t \in T$ au plus quatre fois, avec un coût estimatif total de routage inférieur ou égal à $2\sum_{\{s,t\}\in E(F)} c_{st}y(s,t)$, ce qui est inférieur ou égal à $2c_S(x^*) + 2c_S(x)$ d'après (22.17). Si aucune des opérations n'a un coût estimatif inférieur ou égal à $-\frac{\epsilon}{|\mathcal{F}|}c(x)$, nous avons $-\epsilon c(x) \leq -c_F(x) + 4c_F(x^*) + 2c_S(x^*) + 2c_S(x)$, comme requis. □

Nous pouvons conclure des résultats précédents :

Théorème 22.29. *Soit $0 < \epsilon \leq 1$ et soient x, x^* des solutions réalisables d'une instance donnée et supposons $c^x(t,\delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ pour $t \in \mathcal{F}$ et $\delta \in \mathbb{R}_+$ et $c^x(A,\delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ pour toutes les étoiles et comètes A et $\delta \in \Delta_A^x$. Alors $c(x) \leq (1+\epsilon)(7c_F(x^*) + 5c_S(x^*))$.*

Preuve. D'après le lemme 22.26, nous avons $c_S(x) \leq c_F(x^*) + c_S(x^*) + \frac{\epsilon}{8}c(x)$ et d'après le lemme 22.28, nous avons $c_F(x) \leq 4c_F(x^*) + 2c_S(x^*) + 2c_S(x) + \frac{\epsilon}{8}c(x)$. Ainsi $c(x) = c_F(x) + c_S(x) \leq 7c_F(x^*) + 5c_S(x^*) + \frac{\epsilon}{2}c(x)$, ce qui implique $c(x) \leq (1+\epsilon)(7c_F(x^*) + 5c_S(x^*))$. □

Nous appliquons finalement une technique standard de réduction d'échelle et obtenons le résultat principal de ce paragraphe :

Théorème 22.30. (Vygen [2007]) *Pour tout $\epsilon > 0$, il existe un algorithme d'approximation de facteur $\left(\frac{\sqrt{41}+7}{2} + \epsilon\right)$ polynomial pour le* PROBLÈME DE LOCALISATION UNIVERSEL.

Preuve. Nous pouvons supposer $\epsilon \leq \frac{1}{3}$. Considérons $\beta := \frac{\sqrt{41}-5}{2} \approx 0,7016$. Posons $f_i'(z) := \beta f_i(z)$ pour tout $z \in \mathbb{R}_+$ et $i \in \mathcal{F}$ et considérons l'instance modifiée.

Soit x une solution réalisable de départ. Appliquons les algorithmes du lemme 22.25 et du lemme 22.27 avec $\frac{\epsilon}{16|\mathcal{F}|}$ à la place de ϵ. Soit ils trouvent une opération ADD ou PIVOT, réduisant le coût de la solution courante x d'au moins $\frac{\epsilon}{16|\mathcal{F}|}c(x)$, soit ils concluent que les hypothèses du théorème 22.29 sont satisfaites.

Notons respectivement c_F' et c_F les coûts de construction de l'instance modifiée et de l'instance de départ. Si x est la solution obtenue et x^* est une solution réalisable, alors $c_F(x) + c_S(x) = \frac{1}{\beta}c_F'(x) + c_S(x) \leq \frac{1}{\beta}(6c_F'(x^*) + 4c_S(x^*) + \frac{3\epsilon}{8}c(x)) + c_F'(x^*) + c_S(x^*) + \frac{\epsilon}{8}c(x) \leq (6+\beta)c_F(x^*) + (1+\frac{4}{\beta})c_S(x^*) + \frac{3\epsilon}{4}c(x) = (6+\beta)(c_F(x^*) + c_S(x^*)) + \frac{3\epsilon}{4}c(x)$. Ainsi $c(x) \leq (1+\epsilon)(6+\beta)c(x^*)$.

Chaque itération réduit le coût par un facteur supérieur ou égal à $\frac{1}{1-\frac{\epsilon}{16|\mathcal{F}|}}$, ainsi après $\frac{1}{-\log(1-\frac{\epsilon}{16|\mathcal{F}|})} < \frac{16|\mathcal{F}|}{\epsilon}$ itérations, le coût est au moins réduit par un facteur

égal à 2 (remarquons que $\log x < x - 1$ pour $0 < x < 1$). Cela implique un temps de calcul faiblement polynomial. □

En particulier, comme $\frac{\sqrt{41}+7}{2} < 6,702$, nous avons un algorithme d'approximation de facteur $6,702$. C'est la meilleure garantie de performance actuellement connue.

Exercices

1. Montrer qu'il n'existe pas d'algorithme d'approximation de facteur constant pour le PROBLÈME DU k-MÉDIAN (où l'on ne requiert pas des coûts de service métriques) à moins que *P = NP*.
2. Considérons une instance du PROBLÈME DE LOCALISATION SANS CAPACITÉS. Prouvons que $c_S : 2^{\mathcal{F}} \to \mathbb{R}_+ \cup \{\infty\}$ est supermodulaire, où $c_S(X) := \sum_{j\in\mathcal{D}} \min_{i\in X} c_{ij}$.
3. Considérons une formulation différente comme PL en nombres entiers du PROBLÈME DE LOCALISATION SANS CAPACITÉS, avec une variable z_S de type 0/1 pour chaque paire $S \in \mathcal{F} \times 2^{\mathcal{D}}$:

$$\begin{array}{lll} \min & \displaystyle\sum_{S=(i,D)\in\mathcal{F}\times 2^{\mathcal{D}}} \left(f_i + \sum_{j\in D} c_{ij}\right) z_S & \\ \text{s.c.} & \displaystyle\sum_{S=(i,D)\in\mathcal{F}\times 2^{\mathcal{D}}: j\in D} z_S \geq 1 & (j \in \mathcal{D}) \\ & z_S \in \{0,1\} & (S \in \mathcal{F} \times 2^{\mathcal{D}}). \end{array}$$

   Considérons la relaxation linéaire et son dual. Montrer comment les résoudre en temps polynomial (malgré leur taille exponentielle). Montrer que la valeur optimale du PL est la même que celle de (22.2) et (22.3).
4. Considérons la relaxation linéaire d'un cas particulier simple du PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS, dans lequel chaque installation peut servir jusqu'à u clients ($u \in \mathbb{N}$) : ce PL est obtenu en complétant (22.2) par les contraintes $y_i \leq 1$ et $\sum_{j\in\mathcal{D}} x_{ij} \leq u y_i$ pour $i \in \mathcal{F}$.
   Montrer que cette classe de PLs a un saut intégral non borné, c.-à-d. que le rapport entre le coût d'une solution entière optimale et la valeur optimale du PL peut être arbitrairement grand.
   (Shmoys, Tardos et Aardal [1997])
5. Considérons le PROBLÈME DE LOCALISATION SANS CAPACITÉS avec la propriété qu'à chaque client $j \in \mathcal{D}$ est associé une demande $d_j > 0$ et que les coûts de fonctionnement par unité de demande sont métriques, c.-à-d. $\frac{c_{ij}}{d_j} + \frac{c_{i'j}}{d_j} + \frac{c_{i'j'}}{d_{j'}} \geq \frac{c_{ij'}}{d_{j'}}$ pour $i, i' \in \mathcal{F}$ et $j, j' \in \mathcal{D}$. Modifier les algorithmes d'approximation donnés dans le cas de demandes unitaires et montrer

que les mêmes garanties de performance peuvent être obtenues dans ce cas plus général.

6. Montrer que (22.7) peut en fait être reformulé de manière équivalente comme un PL.

7. Considérons le PL révélateur de facteur (22.7) pour $\gamma_F = 1$. Montrer que le supremum de l'optimum pour tout $d \in \mathbb{N}$ est égal à 2.
(Jain *et al.* [2003])

8. Considérons une instance du PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS. L'objectif est de trouver un ensemble $X \subseteq \mathcal{F}$ de telle sorte que $\sum_{j \in \mathcal{D}} \min_{i \in X} c_{ij}^2$ soit minimum. Trouver un algorithme d'approximation de facteur constant pour ce problème. Essayer d'atteindre un rapport de performance inférieur à 3.

9. Combiner le théorème 22.3 et le théorème 22.10 pour montrer que l'ALGORITHME DE JAIN-VAZIRANI combiné avec la réduction d'échelle et l'augmentation gloutonne a un rapport de performance égal à $1,853$.

∗ 10. Le PROBLÈME DE COUVERTURE-k-MAX est défini de la façon suivante. Étant donné un système d'ensembles $(U, \mathcal{F})$ et un entier k, trouver un sous-ensemble $\mathcal{S} \subseteq \mathcal{F}$ tel que $|\mathcal{S}| = k$ et que $|\bigcup \mathcal{S}|$ soit maximum. Prouver que l'algorithme glouton naturel (sélectionner itérativement un ensemble couvrant autant d'éléments que possible) est un algorithme d'approximation de facteur $(\frac{e}{e-1})$ pour le PROBLÈME DE COUVERTURE-k-MAX.

11. Montrer qu'il existe un algorithme d'approximation de facteur 2 pour le PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS SOUPLES.
Indication : combiner la preuve du théorème 22.22 avec l'analyse de l'ALGORITHME PAR AJUSTEMENT DUAL ; (22.6) peut ici être renforcé.
(Mahdian, Ye et Zhang [2006])

12. Combiner la recherche locale (théorème 22.20) et la discrétisation des coûts (lemme 22.15) à la réduction d'échelle et l'augmentation gloutonne (théorème 22.10) pour obtenir un algorithme d'approximation de facteur $2,375$ pour le PROBLÈME DE LOCALISATION MÉTRIQUE SANS CAPACITÉS.

13. Considérons le cas particulier du PROBLÈME DE LOCALISATION UNIVERSEL où les fonctions coût f_i sont linéaires pour tout $i \in \mathcal{F}$. Décrire un algorithme d'approximation de facteur 3 pour ce cas particulier.

14. Soient $\alpha_0, \alpha_1, \ldots, \alpha_r \in \mathbb{R}_+$ tels que $\alpha_1 = \max_{i=1}^r \alpha_i$ et $S := \sum_{i=0}^r \alpha_i$. Montrer qu'il existe une partition $\{2, \ldots, r\} = I_0 \dot\cup I_1$ telle que $\alpha_k + \sum_{i \in I_k} 2\alpha_i \leq S$ pour $k = 0, 1$.
Indication : trier les éléments de la liste et prendre un élément sur deux.

∗ 15. Considérons un algorithme de recherche locale pour le PROBLÈME DE LOCALISATION MÉTRIQUE AVEC CAPACITÉS qui, par rapport à l'algorithme du paragraphe 22.8, possède une opération PIVOT supplémentaire sur les forêts qui sont l'union disjointe de deux étoiles. On peut prouver que cette opération peut être implémentée pour s'exécuter en temps polynomial dans ce cas particulier.

Montrer qu'avec cette opération supplémentaire on peut obtenir un rapport de performance égal à $5,83$.
Indication : modifier la preuve du lemme 22.28 en utilisant cette nouvelle opération. Utiliser l'exercice 14.
(Zhang, Chen et Ye [2004])

Références

Littérature générale :

Cornuéjols, G., Nemhauser, G.L., Wolsey, L.A. [1990] : The uncapacitated facility location problem. In : Discrete Location Theory (P. Mirchandani, R. Francis, eds.), Wiley, New York 1990, pp. 119–171

Shmoys, D.B. [2000] : Approximation algorithms for facility location problems. Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization ; LNCS 1913 (K. Jansen, S. Khuller, eds.), Springer, Berlin 2000, pp. 27–33

Vygen, J. [2005] : Approximation algorithms for facility location problems (lecture notes). Report No. 05950-OR, Research Institute for Discrete Mathematics, University of Bonn, 2005

Références citées :

Archer, A., Rajagopalan, R., Shmoys, D.B. [2003] : Lagrangian relaxation for the k-median problem : new insights and continuity properties. Algorithms – Proceedings of the 11th Annual European Symposium on Algorithms, Springer, Berlin 2003, pp. 31-42.

Arora, S., Raghavan, P., Rao, S. [1998] : Approximation schemes for Euclidean k-medians and related problems. Proceedings of the 30th Annual ACM Symposium on Theory of Computing (1998), 106–113

Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V. [2004] : Local search heuristics for k-median and facility location problems. SIAM Journal on Computing 33 (2004), 544–562

Balinski, M.L. [1965] : Integer programming : methods, uses, computation. Management Science 12 (1965), 253–313

Balinski, M.L., Wolfe, P. [1963] : On Benders decomposition and a plant location problem. Working paper ARO-27. Mathematica, Princeton 1963

Byrka, J. [2007] : An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems ; LNCS 4627 (M. Charikar, K. Jansen, O. Reingold, J.D.P. Rolim, eds.), Springer, Berlin 2007, pp. 29–43

Byrka, J., Aardal, K. [2007] : The approximation gap for the metric facility location problem is not yet closed. Operations Research Letters 35 (2007), 379–384

Charikar, M., Guha, S. [2005] : Improved combinatorial algorithms for the facility location and k-median problems. SIAM Journal on Computing 34 (2005), 803–824

Charikar, M., Guha, S., Tardos, É., Shmoys, D.B. [2002] : A constant-factor approximation algorithm for the k-median problem. Journal of Computer and System Sciences 65 (2002), 129–149

Chudak, F.A., Shmoys, D.B. [2003] : Improved approximation algorithms for the uncapacitated facility location problem. SIAM Journal on Computing 33 (2003), 1–25

Feige, U. [1998] : A threshold of $\ln n$ for the approximating set cover. Journal of the ACM 45 (1998), 634–652

Guha, S., Khuller, S. [1999] : Greedy strikes back : improved facility location algorithms. Journal of Algorithms 31 (1999), 228–248

Hochbaum, D.S. [1982] : Heuristics for the fixed cost median problem. Mathematical Programming 22 (1982), 148–162

Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.V. [2003] : Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. Journal of the ACM 50 (2003), 795–824

Jain, K., Vazirani, V.V. [2001] : Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. Journal of the ACM 48 (2001), 274–296

Kolliopoulos, S.G., Rao, S. [2007] : A nearly linear-time approximation scheme for the Euclidean k-median problem. SIAM Journal on Computing 37 (2007), 757–782

Korupolu, M., Plaxton, C., Rajaraman, R. [2000] : Analysis of a local search heuristic for facility location problems. Journal of Algorithms 37 (2000), 146–188

Kuehn, A.A., Hamburger, M.J. [1963] : A heuristic program for locating warehouses. Management Science 9 (1963), 643–666

Levi, R., Shmoys, D.B., Swamy, C. [2004] : LP-based approximation algorithms for capacitated facility location. In : Integer Programming and Combinatorial Optimization ; Proceedings of the 10th International IPCO Conference ; LNCS 3064 (G. Nemhauser, D. Bienstock, eds.), Springer, Berlin 2004, pp. 206–218

Mahdian, M., Pál, M. [2003] : Universal facility location. In : Algorithms – Proceedings of the 11th European Symposium on Algorithms (ESA) ; LNCS 2832 (G. di Battista, U. Zwick, eds.), Springer, Berlin 2003, pp. 409–421

Mahdian, M., Ye, Y., Zhang, J. [2006] : Approximation algorithms for metric facility location problems. SIAM Journal on Computing 36 (2006), 411–432

Manne, A.S. [1964] : Plant location under economies-of-scale-decentralization and computation. Management Science 11 (1964), 213–235

Pál, M., Tardos, É., Wexler, T. [2001] : Facility location with hard capacities. Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science (2001), 329–338

Shmoys, D.B., Tardos, É., Aardal, K. [1997] : Approximation algorithms for facility location problems. Proceedings of the 29th Annual ACM Symposium on Theory of Computing (1997), 265–274

Stollsteimer, J.F. [1963] : A working model for plant numbers and locations. Journal of Farm Economics 45 (1963), 631–645

Sviridenko, M. [2002] : An improved approximation algorithm for the metric uncapacitated facility location problem. In : Integer Programming and Combinatorial Optimization ;

Proceedings of the 9th International IPCO Conference ; LNCS 2337 (W. Cook, A. Schulz, eds.), Springer, Berlin 2002, pp. 240–257

Vygen, J. [2007] : From stars to comets : improved local search for universal facility location. Operations Research Letters 35 (2007), 427–433

Zhang, J., Chen, B., Ye, Y. [2004] : Multi-exchange local search algorithm for the capacitated facility location problem. In : Integer Programming and Combinatorial Optimization ; Proceedings of the 10th International IPCO Conference ; LNCS 3064 (G. Nemhauser, D. Bienstock, eds.), Springer, Berlin 2004, pp. 219–233

Zhang, P. [2007] : A new approximation algorithm for the k-facility location problem. Theoretical Computer Science 384 (2007), 126–135

Notations

$\mathbb{N}$	ensemble des entiers naturels $\{1, 2, 3, \ldots\}$	
$\mathbb{Z}$ ($\mathbb{Z}_+$)	ensemble des entiers (non négatifs)	
$\mathbb{Q}$ ($\mathbb{Q}_+$)	ensemble des rationnels (non négatifs)	
$\mathbb{R}$ ($\mathbb{R}_+$)	ensemble des nombres réels (non négatifs)	
$\subset$	inclusion propre	
$\subseteq$	inclusion large	
$\dot{\cup}$	union disjointe	
$X \triangle Y$	différence symétrique des ensembles X et Y	
$\|\|x\|\|_2$	norme euclidienne du vecteur x	
$\|\|x\|\|_\infty$	norme infini du vecteur x	
$x \bmod y$	le nombre unique z tel que $0 \leq z < y$ et $\frac{x-z}{y} \in \mathbb{Z}$	
$x^\top$, $A^\top$	transposé du vecteur x et de la matrice A	
$\lceil x \rceil$	arrondi supérieur de x : plus petit entier supérieur ou égal à x	
$\lfloor x \rfloor$	partie entière de x : plus grand entier inférieur ou égal à x	
$f = O(g)$	notation O	4
$f = \Theta(g)$	notation Θ	4
$\text{taille}(x)$	longueur de la chaîne binaire codant x	6, 74, 376
$\log x$	logarithme en base 2 de x	7
$V(G)$	ensemble des sommets d'un graphe G	13
$E(G)$	ensemble des arêtes (ou des arcs) d'un graphe G	13
$G[X]$	sous-graphe de G induit par $X \subseteq V(G)$	14
$G - v$	sous-graphe de G induit par $V(G) \setminus \{v\}$	14
$G - e$	graphe obtenu par suppression de l'arête (ou de l'arc e) de G	14
$G + e$	graphe obtenu en ajoutant l'arête (ou l'arc) e à G	14
$G + H$	somme des graphes G et H	14
G/X	graphe résultant de G après contraction de l'ensemble $X \subseteq V(G)$	14
$E(X, Y)$	ensemble des arêtes incidentes à $X \setminus Y$ et à $Y \setminus X$	14
$E^+(X, Y)$	ensemble des arcs sortant de $X \setminus Y$ et entrant dans $Y \setminus X$	14
$\delta(X)$, $\delta(v)$	$E(X, V(G) \setminus X)$, $E(\{v\}, V(G) \setminus \{v\})$	14
$\Gamma(X)$, $\Gamma(v)$	ensemble des voisins de l'ensemble $X \subseteq V(G)$, ou du sommet v	14
$\delta^+(X)$, $\delta^+(v)$	ensemble des arcs sortant de l'ensemble $X \subseteq V(G)$, du sommet v	14
$\delta^-(X)$, $\delta^-(v)$	ensemble des arcs entrant dans l'ensemble $X \subseteq V(G)$, dans le sommet v	14
2^S	ensemble des parties de S	15

K_n	graphe complet sur n sommets	15
$P_{[x,y]}$	sous-chaîne (ou sous-chemin) de P de x à y	16
$\mathrm{dist}(v,w)$	longueur d'une plus courte chaîne (ou chemin) de x à y	17
$c(F)$	$\sum_{e\in F} c(e)$ (en supposant $c : E \to \mathbb{R}$ et $F \subseteq E$)	17
$K_{n,m}$	graphe biparti complet sur n et m sommets	34
$cr(J,l)$	nombre de croisements du polygone J avec la ligne l	36, 565
G^*	graphe planaire dual de G	42
e^*	arête de G^* duale de e	42
$x^\top y$, xy	produit scalaire des vecteurs x et y	51
$x \leq y$	les composantes du vecteur $y - x$ sont non négatives	51
$\mathrm{rang}(A)$	rang de la matrice A	53
$\dim X$	dimension d'un ensemble non vide $X \subseteq \mathbb{R}^n$	53
I	matrice identité	55
e_j	j-ième vecteur unité	55
A_J	sous-matrice de A restreinte aux lignes indicées par J	56
b_J	sous-vecteur de b restreint aux composantes indicées par J	56
$\mathbb{1}$	vecteur dont toutes les composantes valent un	59
A^J	sous-matrice de A restreinte aux colonnes indicées par J	60
$\mathrm{conv}(X)$	enveloppe convexe de l'ensemble de tous les vecteurs de X	67
$\det A$	déterminant de la matrice A	75
$\mathrm{sgn}(\pi)$	signature de la permutation π	75
$E(A,x)$	ellipsoïde	83
$B(x,r)$	boule euclidienne de centre x et de rayon r	83
$\mathrm{volume}\,(X)$	volume de l'ensemble non vide $X \subseteq \mathbb{R}^n$	83
$\|\|A\|\|$	norme de la matrice A	85
X°	ensemble polaire de X	95
P_I	enveloppe convexe entière du polyèdre P	101
$\Xi(A)$	valeur absolue maximum des sous-déterminants de la matrice A	103
P', $P^{(i)}$	première, i-ième troncature de Gomory-Chvátal de P	118
$LR(\lambda)$	relaxation lagrangienne	122
$\delta(X_1,\ldots,X_p)$	multicoupe	146
$c_\pi((x,y))$	coût réduit de l'arc (x,y) déduit des potentiels π	160
$(\bar{G},\bar{c})$	fermeture métrique de (G,c)	161
$\mathrm{ex}_f(v)$	différence entre le flot entrant et le flot sortant en v	171
$\mathrm{valeur}\,(f)$	valeur d'un flot f de s à t	171
$\overleftrightarrow{G}$	graphe obtenu en ajoutant à G les arcs inversés	173
$\overleftarrow{e}$	arc inversé d'un arc e	173
$u_f(e)$	capacité résiduelle d'un arc e associée au flot f	173
G_f	graphe résiduel associé au flot f	173
G_f^L	graphe niveau de G_f	180
λ_{st}	capacité minimum d'une coupe séparant s et t	187
$\lambda(G)$	capacité minimum d'une coupe de G (arête-connexité)	195

$\nu(G)$	cardinalité maximum d'un couplage de G	236
$\tau(G)$	cardinalité minimum d'une couverture par les sommets de G	236
$T_G(x)$	matrice de Tutte de G, associée au vecteur x	238
$q_G(X)$	nombre de composantes connexes impaires de $G - X$	241
$\alpha(G)$	cardinalité maximum d'un ensemble stable maximum de G	259
$\zeta(G)$	cardinalité minimum d'une couverture par les arêtes de G	259
$r(X)$	rang de X dans un système d'indépendance	316
$\sigma(X)$	fermeture de X dans un système d'indépendance	316
$\mathcal{M}(G)$	matroïde graphique d'un graphe G non orienté	318
$\rho(X)$	rang inférieur de X dans un système d'indépendance	319
$q(E, \mathcal{F})$	rang quotient d'un système d'indépendance $(E, \mathcal{F})$	319
$C(X, e)$	si $X \in \mathcal{F}$: le circuit unique contenu dans $X \cup \{e\}$ (ou $\emptyset$ si $X \cup \{e\} \in \mathcal{F}$)	324
$(E, \mathcal{F}^*)$	dual d'un système d'indépendance $(E, \mathcal{F})$	325
$P(f)$	polymatroïde associé à une fonction sous-modulaire f	355
$\sqcup$	symbole blanc	376
$\{0,1\}^*$	ensemble de toutes les chaînes binaires	376
P	classe des problèmes de décision résolubles en temps polynomial	384
NP	classe des problèmes de décision avec des certificats vérifiant les instances-«oui»	385
$\overline{x}$	négation du littéral x	388
coNP	classe complémentaire de *NP*	400
$\mathrm{OPT}(x)$	valeur d'une solution optimale pour l'instance x	403
$A(x)$	valeur de l'output de l'algorithme A pour un problème d'optimisation, l'input étant x	403
$\mathrm{large}(x)$	plus grand entier apparaissant dans l'input x	404
$H(n)$	$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$	415
$\chi(G)$	nombre chromatique de G	430
$\omega(G)$	cardinalité maximum d'une clique de G	430
$\mathrm{Exp}(X)$	espérance d'une variable aléatoire X	438
$\mathrm{Prob}(X)$	probabilité d'un événement X	438
$\mathrm{SUM}(I)$	somme sur tous les éléments de I	475
$NF(I)$	output de l'ALGORITHME NEXT-FIT sur l'instance I	478
$FF(I)$	output de l'ALGORITHME FIRST-FIT sur l'instance I	478
$FFD(I)$	output de l'ALGORITHME FIRST-FIT DÉCROISSANT I sur l'instance I	480
$G_i^{(a,b)}$	grille déplacée	564
$Q(n)$	enveloppe convexe des vecteurs d'incidence des tours de K_n	577
$HK(K_n, c)$	borne de Held et Karp pour l'instance (K_n, c) du TSP	584
$c_F(X)$, $c_F(x)$	coût des services dans une solution	599, 621
$c_S(X)$, $c_S(x)$	coût des usagers dans une solution	599, 621

Index des noms d'auteurs

Index général