Vijay V. Vazirani

College of Computing
Georgia Institute of Technology

Traduction de Nicolas Schabanel
CNRS, École Normale Supérieure de Lyon

Algorithmes d'approximation

Springer
Berlin Heidelberg New York
Barcelona Hong Kong
London Milan Paris
Singapore Tokyo

À mes parents

« Cet ouvrage aborde les développements théoriques majeurs permettant d'obtenir des solutions approchées aux problèmes difficiles d'optimisation combinatoire ou de dénombrement. Il comporte une présentation élégante de la théorie combinatoire, de nombreux algorithmes aussi utiles qu'intéressants et des résultats remarquables concernant la complexité des problèmes traités. La clarté de l'exposé et la sélection judicieuse des exercices rendent cet ouvrage attractif et accessible pour tous les lecteurs ayant un goût pour les mathématiques et l'algorithmique. »

Richard Karp, Professeur à l'Université de Californie, Berkeley

« Suite au développement des techniques fondamentales d'optimisation combinatoire des années 1960 et 1970, une question importante consistait à développer une théorie algorithmique de l'approximation. Dans les années 1990, les progrès conjoints réalisés tant dans la conception d'algorithmes d'approximation que dans la découverte de preuves établissant la complexité de l'approximation de certains problèmes, ont conduit à une théorie élégante. La nécessité de résoudre des instances de plus en plus grandes pour des problèmes difficiles, comme ceux posés par Internet ou la génomique, a renforcé l'intérêt pour cette théorie. Il s'agit d'un domaine de recherche très actif qui voit sa boîte à outils s'enrichir de jour en jour.

C'est un plaisir de recommander l'ouvrage de Vijay Vazirani, complet et bien écrit, qui traite de ce sujet d'actualité. Je suis certain qu'il sera très utile au lecteur qui y trouvera aussi bien une introduction à l'algorithmique de l'approximation qu'un texte de référence sur de nombreux aspects du domaine. »

László Lovász, Directeur de recherche, Microsoft Recherche

Préface

Bien que cela puisse paraître paradoxal, toute science exacte est dominée par la notion d'approximation.
Bertrand Russell (1872-1970)

La plupart des problèmes d'optimisation naturels sont **NP**-difficiles, et tout particulièrement ceux qui ont des applications réelles importantes. Suivant la conjecture largement admise $\mathbf{P} \neq \mathbf{NP}$, leur résolution exacte impliquerait un temps de calcul prohibitif. Caractériser la difficulté de l'approximation de ces problèmes par des algorithmes de temps polynomial est donc un sujet d'étude inévitable en informatique et en mathématiques. Cet ouvrage dresse un tableau de la théorie de l'algorithmique d'approximation à ce jour. Il est raisonnable de penser que cette image évoluera dans le temps.

L'exposé se divise en trois parties. La première partie présente des algorithmes d'approximation combinatoires pour des problèmes fondamentaux en mettant en œuvre une grande diversité de techniques algorithmiques. La diversité de ces techniques peut paraître déconcertante au premier abord. La nature est en effet riche et nous ne pouvons pas espérer qu'un nombre limité d'astuces permette de résoudre la très grande variété des problèmes **NP**-difficiles. Nous avons volontairement évité de trop catégoriser les différentes techniques, afin de ne pas restreindre leurs portées. Au contraire, nous avons tenté d'extraire aussi précisément que possible les caractéristiques individuelles de chaque problème, ainsi que les relations entre ces problèmes et les solutions algorithmiques proposées.

La deuxième partie est consacrée aux approximations reposant sur la programmation linéaire. Deux techniques fondamentales y sont présentées : la méthode de l'arrondi et la méthode primal-dual. La qualité de la solution approchée dépend principalement du programme linéaire choisi et de sa relaxation. Il n'y a pas de recette miracle pour trouver une bonne relaxation d'un problème, de même qu'il n'en existe pas pour démontrer un théorème en mathématiques (les lecteurs familiers avec la théorie de la complexité reconnaîtront la question sous-jacente $\mathbf{P} \neq \mathbf{NP}$).

La dernière partie détaille quatre thématiques importantes. La première est la recherche d'un vecteur non nul le plus court dans un module (chapitre 27). Différentes raisons font que ce problème mérite d'être traité à part.

La deuxième thématique étudie le dénombrement approché des solutions d'un problème, ce qui est une question très différente de celle de l'approximation de la valeur d'une solution d'un problème d'optimisation. Le dénombrement des solutions de presque tous les problèmes **NP**-complets connus est #**P**-complet. Il est intéressant de constater qu'à l'exception d'une poignée de problèmes, c'est également vrai pour les problèmes de **P**. Une théorie très avancée a été proposée pour obtenir des algorithmes de dénombrement efficaces pour les problèmes de **P**. La plupart de ces algorithmes utilisent des méthodes de Monte Carlo à base de chaînes de Markov (MCMC), un sujet qui nécessiterait un livre à lui tout seul et qui ne sera donc pas traité ici. Le chapitre 28 présente deux solutions algorithmiques combinatoires, qui n'utilisent pas la méthode MCMC, pour deux problèmes fondamentaux de dénombrement.

La troisième thématique est consacrée aux résultats récents qui ont confirmé le fondement d'une théorie propre de l'algorithmique d'approximation en quantifiant la difficulté de l'approximation pour plusieurs problèmes clés. Le chapitre 29 passe en revue ces résultats. Le principal point technique, le théorème PCP, y est admis. Nous ne connaissons malheureusement pas de preuve simple de ce théorème à l'heure actuelle.

Nous avons regroupé dans le dernier thème de nombreux problèmes ouverts de ce jeune champ de recherche. Cette liste n'est pas exhaustive. Elle est plutôt centrée autour des sujets d'étude actuellement actifs. On a cherché, durant quarante ans, des algorithmes exacts pour ces problèmes, sans réelle avancée. Étant donné que parmi les problèmes naturels, les algorithmes polynomiaux sont plutôt l'exception que la règle, il est raisonnable de croire que l'importance de l'algorithmique d'approximation va croître considérablement dans les années à venir.

Le problème de couverture par ensembles minimum occupe une place particulière dans la théorie de l'approximation et donc dans cet ouvrage. Sa définition très simple permet d'introduire à la fois des concepts clés et quelques techniques algorithmiques de base des parties I et II. La troisième partie complète le traitement de ce problème central par un résultat démontrant la difficulté de son approximation, dont la preuve est assez élaborée. La borne donnée par ce résultat est asymptotiquement égale à celle du meilleur algorithme d'approximation pour ce problème — ce qui est une raison supplémentaire de présenter cette preuve plutôt difficile.

Le travail du sculpteur Michel-Ange est un modèle pour notre approche de la conception et de la présentation des algorithmes. Une part essentielle du travail du sculpteur consiste en effet à étudier les pierres intéressantes d'une carrière, pour en découvrir les formes qu'elles épousent naturellement. Puis, le travail au ciseau consiste à suivre ces formes de manière naturelle, ou encore minimale. Par analogie, nous commençons par étudier un problème formulé

clairement et simplement (peut-être une version simplifiée du problème original). La plus grande partie du travail d'un algorithmicien est de comprendre quelle est la structure combinatoire déterminante sur le plan algorithmique. L'algorithme s'articule ensuite autour de cette structure de façon minimale. La présentation des algorithmes s'inscrit dans cette analogie, en insistant sur la découverte des structures des problèmes puis en présentant les algorithmes sous une forme minimale.

Nous avons essayé de rédiger chaque chapitre de façon concise et simple, quitte à ne présenter souvent que le résultat principal. Les généralisations et résultats similaires sont laissés en exercices. Nous avons également été amenés à proposer en exercices des résultats importants qu'il n'a pas été possible de développer en détail dans ce volume. Des indications sont données pour certains exercices sans pour autant que cela soit significatif de leur difficulté.

Cet ouvrage correspond à un ou plusieurs cours d'algorithmes d'approximation de fin de licence ou de mastère. Il contient plus du double de ce qui peut être présenté en un semestre, permettant ainsi à l'enseignant de choisir librement les sujets abordés. Un cours de licence d'introduction à l'algorithmique, à la théorie des graphes et sur la théorie de la **NP**-complétude devrait être un prérequis suffisant pour la plupart des chapitres. Par souci de complétude, le lecteur trouvera en annexe divers rappels des principaux résultats et définitions utilisés : théorie de la complexité en annexe A, probabilités en annexe B, programmation linéaire au chapitre 12, programmation semi-définie au chapitre 26, modules au chapitre 27. Nous avons choisi d'accorder une part sans doute disproportionnée à l'autoréductibilité dans l'annexe A car cette notion est rarement traitée ailleurs. Cet ouvrage peut également compléter un cours d'algorithmique fondamentale en 1^{er} ou $2^{\text{ème}}$ cycle (tout particulièrement les premiers chapitres des parties I et II). Les chapitres de ces deux parties sont de difficulté croissante.

Afin de toucher un public aussi large que possible, nous avons décidé de ne publier cet ouvrage dans aucune des séries spéciales de Springer — pas même la prestigieuse série jaune (nous n'avons cependant pas pu résister à l'ajout d'une touche de jaune sur la couverture de l'édition originale). Tous les commentaires et toutes les corrections sont les bienvenus. Nous avons créé une adresse électronique à cet effet pour l'édition en langue française : algo_approx@gmail.com[1].

Enfin, quelques mots à propos de l'impact concret de ces résultats. Sachant qu'en pratique les gens ont les yeux rivés sur des marges d'erreur entre 2 % et 5 %, quelle est l'utilité d'algorithmes à un facteur 2, voire à $O(\log n)$ de l'optimal ? De même, quel est l'intérêt d'améliorer la qualité de l'approximation d'un facteur 2 à 3/2 ?

Considérons ces deux questions en explicitant tout d'abord leur caractère fallacieux. La garantie de performance est une borne sur la performance

[1] L'adresse électronique dédiée aux remarques concernant l'édition originale est : approx@cc.gatech.edu.

de l'algorithme pour la pire instance. Il vaut peut-être mieux voir le facteur d'approximation comme une mesure qui encourage à explorer plus en détails la structure combinatoire du problème, afin de découvrir des outils plus puissants pour l'exploiter. Il est reconnu qu'il est d'autant plus difficile de trouver des instances critiques atteignant les bornes que les algorithmes offrent de meilleures garanties. Pour certains algorithmes récents, l'obtention de telles instances a fait l'objet d'un article à elle seule (par exemple, voir section 26.7). Des expérimentations ont confirmé que ces algorithmes sophistiqués ont la marge d'erreur souhaitée, de 2 % à 5 %, sur les instances typiques rencontrées en pratique, même si leur pire cas est bien au-delà. Enfin, les algorithmes prouvés théoriquement doivent être considérés comme une base algorithmique qu'il faut adapter soigneusement aux instances se présentant dans des applications spécifiques.

Nous espérons que ce livre catalysera la croissance de cette théorie et de son impact en pratique.

Remerciements

Ce livre s'appuie sur les cours que j'ai donnés à l'Indian Institute of Technology de Delhi durant les sessions de printemps 1992 et 1993, à Georgia Tech durant les sessions de printemps 1997, 1999 et 2000 et à DIMACS durant l'automne 1998. Les notes de cours du printemps 1992 constituèrent la première pierre de cet ouvrage. Il est intéressant de remarquer que plus de la moitié de notre propos repose sur des résultats de recherche issus des années qui ont suivi.

De très nombreux amis — et membres de ma famille — m'ont aidé. Je souhaite tout d'abord remercier Naveen Garg, Kamal Jain, Ion Măndoiu, Sridhar Rajagopalan, Huzur Saran et Mihalis Yannakakis — mes collaborations intenses avec eux m'ont permis de formuler les idées présentées dans cet ouvrage. J'ai eu la chance de bénéficier de l'aide et des conseils de Ion Măndoiu sur de nombreux points — ses qualités artistiques furent d'un grand secours pour la présentation du document et de ses figures. Merci encore, Ion !

Je souhaite exprimer ma gratitude envers les très nombreux experts du domaine pour leur aide généreuse sur le choix des thèmes à aborder et leur présentation, pour leurs commentaires sur les manuscrits, pour avoir assuré la correction et la complétude des références, ainsi que pour la conception des exercices et de la liste des problèmes ouverts. Merci à Sanjeev Arora, Alan Frieze, Naveen Garg, Michel Goemans, Mark Jerrum, Claire Kenyon, Samir Khuller, Daniele Micciancio, Yuval Rabani, Sridhar Rajagopalan, Dana Randall, Tim Roughgarden, Amin Saberi, Leonard Schulman, Amin Shokrollahi, et Mihalis Yannakakis. Un merci tout particulier à Kamal Jain, Éva Tardos, et Luca Trevisan.

De nombreuses autres personnes m'ont aidé par la pertinence de leurs commentaires. Je souhaite remercier Sarmad Abbasi, Cristina Bazgan, Ro-

gerio Brito Gruia Calinescu, Amit Chakrabarti, Mosses Charikar, Joseph Cheriyan, Vasek Chvátal, Uri Feige, Cristina Fernandes, Ashish Goel, Parikshit Gopalan, Mike Grigoriadis, Sudipto Guha, Dorit Hochbaum, Howard Karloff, Leonid Khachian, Stavros Kolliopoulos, Jan van Leeuwen, Nati Lenial, George Leuker, Vangelis Markakis, Aranyak Mehta, Rajeev Motwani, Prabhakar Raghavan, Satish Rao, Miklos Santha, Jiri Sgall, David Shmoys, Alistair Sinclair, Prasad Tetali, Pete Veinott, Ramarathnam Venkatesan, Nisheeth Vishnoi, et David Williamson. J'adresse mes excuses et mes remerciements aux personnes que j'aurais oubliées. De nombreux étudiants ont joué un rôle important, notamment en me confiant les notes qu'ils avaient prises dans mes cours. Je voudrais leur exprimer ma gratitude.

Je souhaite remercier l'IIT Delhi — remerciements particuliers à Shachin Maheshwari — Georgia Tech et DIMACS pour m'avoir fourni des environnements de travail agréables, motivants et riches. Merci à la NSF pour ses dotations financières CCR-9627308 et CCR-9820896.

J'ai particulièrement apprécié de travailler avec Hans Wössner à l'édition de ce livre. Le soin qu'il accorda à ce projet et à son auteur furent impressionnants. Merci encore à Frank Holzwarth pour avoir partagé son expertise en LaTeX.

Un projet de cette importance aurait été difficile à mener à bien sans le soutien sans réserve des membres de ma famille. Heureusement pour moi, quelques-uns sont également des chercheurs — ma femme, Milena Mihail, et mon frère, Umesh Vazirani. L'arrivée de Little Michel à mi-chemin de ce projet m'a apporté de nouvelles joies et énergies, et a rendu ce projet encore plus audacieux ! Je souhaite remercier avant tout mes parents pour leur soutien indéfectible et leur inspiration — mon père, auteur distingué de plusieurs ouvrages d'ingénierie civile, et ma mère, qui connaît si bien la musique classique indienne. Ce livre leur est dédicacé.

Atlanta, Géorgie, mai 2001 Vijay Vazirani

Remerciements du traducteur

Je tiens à remercier Vincent Bouchitté, Maxime Crochemore, Marianne Delorme, Marc Demange, Paul Feautrier, Fabien Feschet, Pierre Fraigniaud, Cyril Gavoille, Isabelle Guérin-Lassous, Olivier Hudry, Claire Kenyon, Pascal Koiran, Emmanuelle Lebhar, Laurent Lyaudet, Jacques Mazoyer, Alain Maruani, Jérôme Monnot, Bruno Petazzoni, Nicolas Puech, Mathieu Raffinot, André Raspaud, Guillaume Theyssier, Arnaud Tisserand, Ioan Todinca, et bien d'autres que j'aurais pu oublier, pour leurs aides, relectures et encouragements lors de cette traduction.

Lyon, France, novembre 2005 Nicolas Schabanel

Sommaire

1 Introduction

Les problèmes d'optimisation **NP**-difficiles forment un ensemble très riche de possibilités : de la possibilité d'approcher avec une précision arbitraire, à l'impossibilité de toute garantie sur la qualité de l'approximation. Malgré cette diversité, la conception d'algorithmes d'approximation suit des principes généraux, que nous allons explorer dans ce chapitre.

Un problème d'optimisation ne se résout en temps polynomial que s'il possède une structure combinatoire pertinente algorithmiquement, qui offre une « prise » pour trouver efficacement une solution optimale. La conception d'un algorithme polynomial exact s'exécute en deux phases : démêler une structure utile du problème, puis trouver un algorithme qui puisse l'exploiter.

Bien que les problèmes **NP**-difficiles n'offrent pas de prises pour trouver une solution optimale efficacement, ils peuvent parfois offrir des prises pour trouver une solution quasi optimale. Ainsi, la conception d'algorithmes d'approximation n'est pas, à première vue, très différente de la conception d'algorithmes exacts. Il s'agit également de démêler une structure pertinente du problème et de trouver un algorithme pour l'exploiter efficacement. Mais, en général, cette structure est plus élaborée, et souvent les techniques algorithmiques utilisées sont obtenues par généralisation ou extension de plusieurs outils puissants développés pour l'étude des algorithmes exacts.

La conception d'algorithmes d'approximation suit cependant ses propres principes. Nous illustrons quelques-uns de ces principes dans la section 1.1, pour le problème suivant.

Problème 1.1 (Couverture par sommets)[1] Étant donné un graphe non orienté $G = (V, E)$ et une fonction de coût sur les sommets $c : V \to \mathbf{Q}^+$, trouver une *couverture par sommets de coût minimal*, c'est-à-dire un sous-ensemble $S \subseteq V$ tel que toutes les arêtes ont (au moins) une extrémité dans S. Le cas particulier où tous les sommets ont un coût unitaire sera appelé le problème de la *couverture par sommets de taille minimum*.[2]

La conception d'algorithmes d'approximation implique une attaque délicate de la **NP**-difficulté afin d'en extraire une solution approchée efficacement. L'annexe A et plusieurs exercices de la section 1.3 présentent des rappels utiles sur les concepts clés de la théorie de la complexité.

[1] *Vertex cover*, en anglais.

[2] *Cardinality vertex cover problem*, en anglais.

Les définitions précises des notions de problème d'optimisation **NP** et d'algorithme d'approximation qui le résout (par exemple, voir exercices 1.9 et 1.10) sont importantes mais techniques ; nous avons donc préféré les donner en annexe A. Le lecteur en retrouvera cependant l'essentiel ci-dessous.

Un problème d'optimisation **NP**, Π, est un problème de minimisation ou de maximisation. À chaque instance valide I de Π, est associé un ensemble de solutions réalisables. À chaque solution réalisable est associé un nombre rationnel, appelé sa *valeur objectif*, ou encore la valeur de la *fonction objectif* en cette solution. Pour un problème d'optimisation **NP**, il existe des algorithmes polynomiaux (en la taille du codage de I et de la solution testée) qui décident de la validité et de la réalisabilité d'une solution et qui en calcule la valeur objectif. Une solution réalisable qui a la valeur optimale de la fonction objectif est appelée *solution optimale*. $\mathrm{OPT}_\Pi(I)$ désigne la valeur objectif d'une solution optimale d'une instance I. Nous l'abrégerons par OPT en absence d'ambiguïté. Le calcul de $\mathrm{OPT}_\Pi(I)$, pour les problèmes étudiés dans cet ouvrage, est **NP**-difficile.

Par exemple, une instance valide pour la couverture par sommets consiste en un graphe non orienté $G = (V, E)$ muni d'une fonction de coût sur ses sommets. Une solution réalisable est un sous-ensemble $S \subseteq V$ qui couvre les arêtes G. Sa valeur objectif est la somme des coûts des sommets de S. Un tel ensemble de coût minimum est une solution optimale.

Un algorithme d'approximation $\mathcal{A}$ pour Π calcule en temps polynomial (en la taille de l'instance) une solution réalisable dont la valeur objectif est « proche » de l'optimal ; par « proche », on entend que cette valeur est à un facteur garanti de la valeur optimale. Dans la section suivante, nous présentons un algorithme polynomial de facteur d'approximation 2 pour le problème de la couverture par sommets de taille minimum, c'est-à-dire un algorithme qui trouve une couverture par sommets de coût inférieur à $2 \cdot \mathrm{OPT}$ en temps polynomial en $|V|$ (le nombre d'arêtes, et donc la taille d'une instance, sont des polynômes en $|V|$). Un tel algorithme polynomial sera appelé une 2-approximation.

1.1 Minorer OPT

La conception d'un algorithme d'approximation pour un problème d'optimisation **NP** et **NP**-difficile fait face immédiatement au dilemme suivant. Le coût de la solution produite doit être comparé au coût optimal afin d'établir la garantie de performance. Or, pour ces problèmes, non seulement le calcul d'une solution optimale est **NP**-difficile, mais le calcul du coût optimal l'est aussi (voir annexe A). Nous verrons en fait, section A.5, que le calcul du coût optimal (ou la résolution du problème de décision associé) est au cœur de la résolution de ces problèmes. Comment alors établir la garantie de performance ? Il est intéressant de noter que la réponse à cette question est une étape clé de la conception d'algorithmes d'approximation.

Prenons l'exemple de la couverture par sommets de taille minimum. Nous contournons la difficulté ci-dessus en construisant un « bon » *minorant*[3] du coût optimal (la taille minimum d'une couverture par sommets), qui, lui, sera calculable en temps polynomial.

1.1.1 Un algorithme d'approximation pour la couverture par sommets de taille minimum

Commençons par quelques définitions. Étant donné $H = (U, F)$, un graphe, on appelle *couplage* de H, tout sous-ensemble d'arêtes $M \subseteq F$ tel qu'aucune paire d'arêtes de M n'a d'extrémité commune. On appelle *couplage maximum*, tout couplage de taille maximum dans H, et *couplage maximal*, tout couplage qui est maximal pour l'inclusion[4]. L'algorithme glouton suivant calcule en temps polynomial un couplage maximal : tant qu'il existe des arêtes, sélectionner une arête et retirer ses extrémités du graphe. Des algorithmes plus sophistiqués permettent également de calculer un couplage maximum en temps polynomial.

Observons que la taille d'un couplage maximal de G est un minorant pour la couverture par sommets de taille minimum. La raison en est que *toute* couverture par sommets doit contenir au moins l'une des extrémités de chaque arête du couplage. La construction de ce minorant suggère immédiatement l'algorithme très simple suivant :

Algorithme 1.2 (Couverture par sommets de taille minimum)
Calculer un couplage maximal de G et renvoyer l'ensemble des extrémités des arêtes du couplage.

Théorème 1.3 *L'algorithme 1.2 est une* 2*-approximation pour le problème de la couverture par sommets de taille minimum.*

Preuve : Toutes les arêtes sont couvertes par l'ensemble de sommets renvoyé — sinon, une arête non couverte pourrait être ajoutée au couplage, qui ne serait pas maximal. Soit M le couplage sélectionné. Nous savons que $|M| \leqslant \mathrm{OPT}$. Le facteur d'approximation se déduit donc ainsi : la taille de la couverture renvoyée est $2\,|M|$, c'est-à-dire $\leqslant 2 \cdot \mathrm{OPT}$. □

[3] *A good lower bound*, en anglais.

[4] Les termes « minimum » et « minimal » réfèrent à deux notions distinctes sur les ordres : un élément minimum d'un ordre est un élément qui est inférieur à tous les éléments de l'ordre, alors qu'un élément minimal est un élément tel qu'aucun autre élément ne lui soit inférieur. Un élément minimum est en particulier minimal, mais la réciproque n'est pas vraie en général (par exemple, elle est vraie pour les ordres totaux). La même remarque s'applique aux termes « maximum » et « maximal ».

Notons que l'algorithme d'approximation découle naturellement de la construction du minorant. Cette situation est typique en algorithmique d'approximation. Nous verrons, partie II, comment la programmation linéaire permet d'unifier la construction de minorants de plusieurs problèmes fondamentaux ; les algorithmes d'approximation se construisent alors directement à partir du programme linéaire qui calcule le minorant.

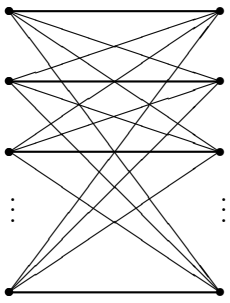
1.1.2 Peut-on améliorer la garantie de performance ?

Plusieurs questions se posent lorsque l'on cherche à améliorer la garantie de performance pour la couverture par sommets de taille minimum :

1. Peut-on améliorer le facteur d'approximation de l'algorithme 1.2 par une meilleure analyse ?
2. Peut-on construire un algorithme avec une meilleure garantie en utilisant le même minorant que l'algorithme 1.2, c'est-à-dire la taille d'un couplage maximum de G ?
3. Existe-t-il une autre construction de minorant qui améliore le facteur d'approximation pour la couverture par sommets ?

L'exemple 1.4 montre que la réponse à la première question est « Non », c'est-à-dire l'analyse de l'algorithme 1.2 donnée ci-dessus est *au plus près*. Il donne une famille infinie d'instances pour lesquelles le coût de la solution proposée par l'algorithme 1.2 est le double de l'optimal. Une famille infinie d'instances de ce type, montrant que l'analyse d'un algorithme d'approximation est au plus près, sera appelée une *instance critique*.[5] La construction d'instances critiques est particulièrement importante. Ces dernières permettent de mieux comprendre les faiblesses d'un algorithme et ont souvent permis d'obtenir des algorithmes avec de meilleures garanties. Nous conseillons vivement au lecteur d'exécuter les algorithmes de ce livre sur leurs instances critiques.

Exemple 1.4 Considérons la famille infinie d'instances formées des graphes bipartis complets $K_{n,n}$.



[5] *Tight analysis* et *tight instance*, en anglais.

L'algorithme 1.2 sélectionne l'ensemble des $2n$ sommets de $K(n,n)$, alors que chaque côté du graphe biparti forme une couverture de taille n. □

Supposons que les facteurs d'approximation soient toujours analysés en comparant le coût de la solution calculée au minorant. C'est en fait le cas de la plupart des algorithmes d'approximation connus. Alors, la réponse à la deuxième question est « Non » également. L'exemple 1.5 exhibe une famille infinie d'instances pour lesquelles le minorant, la taille d'un couplage maximal, vaut la moitié de la taille de la couverture minimale. Dans le cas d'algorithmes fondés sur un programme linéaire, la question analogue se pose en terme d'une quantité fondamentale en programmation linéaire, le saut intégral[6] (voir chapitre 12).

La troisième et dernière question, améliorer la garantie de performance pour la couverture par sommets, est en fait un problème ouvert central de l'algorithmique d'approximation actuellement (voir section 30.1).

Exemple 1.5 Pour la famille infinie d'instances $(K_n)_{n \text{ impair}}$ (les cliques de taille impaire), le minorant, la taille d'un couplage maximal, vaut la moitié de la taille minimale d'une couverture par sommets. Tout couplage maximal compte exactement $(n-1)/2$ arêtes, alors que la taille d'une couverture minimale est $n-1$. □

1.2 Problèmes bien caractérisés et relations min-max

Considérons les problèmes de décision associés aux problèmes de la couverture par sommets de taille minimum et du couplage maximum :

– la taille d'une couverture minimum de G est-elle $\leqslant k$?

– la taille d'un couplage maximum de G est-elle $\geqslant \ell$?

Ces deux problèmes de décision sont dans **NP** et ont donc des certificats positifs (définitions en annexe A). Ces problèmes ont-ils des certificats négatifs ? Nous savons que la taille d'un couplage maximum est un minorant de la taille d'une couverture par sommets minimum. Si G est biparti, on a en fait l'égalité ; c'est le théorème classique de König-Egerváry.

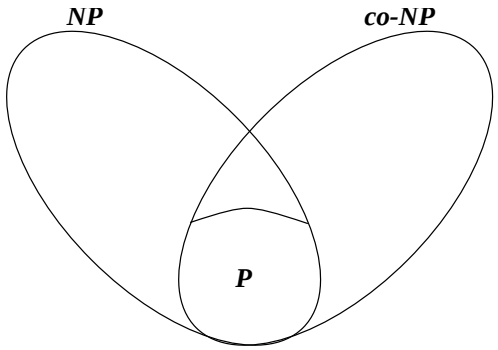
Théorème 1.6 *Pour tout graphe biparti,*

$$\max_{\textit{couplage } M} |M| = \min_{\textit{couverture par sommets } C} |C|$$

Ainsi, pour tout graphe G biparti, si la réponse au premier problème est « Non », il existe dans G un couplage de taille $k+1$ qui est un certificat. De même, si la réponse au second problème est « Non », il existe une couverture de taille $\ell - 1$ dans G. Par conséquent, les restrictions de ces deux problèmes aux graphes bipartis admettent des certificats négatifs et appartiennent à

[6] *Integrality gap*, en anglais.

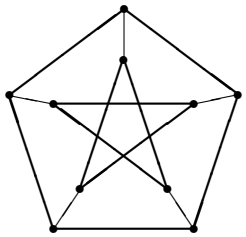
co-**NP**. En fait, ces deux restrictions sont dans **P**. Il est facile de voir que tout problème de **P** admet des certificats positifs et négatifs (la chaîne vide suffit). C'est l'inclusion suivante : $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co-}\mathbf{NP}$. Le consensus actuel estime que cette inclusion est stricte ; cette conjecture est représentée ci-dessous.



Les problèmes qui ont des certificats positifs et négatifs, c'est-à-dire qui sont dans $\mathbf{NP} \cap \text{co-}\mathbf{NP}$, sont dits *bien caractérisés*. Cette notion est importante ; par exemple, c'est en remarquant que le problème du couplage maximum est bien caractérisé que la quête d'un algorithme polynomial pour ce problème a commencé.

Des relations min-max de ce type permettent de prouver qu'un problème est bien caractérisé. Ces relations sont parmi les résultats les plus élégants et les plus puissants de la combinatoire. Plusieurs algorithmes polynomiaux (exacts) fondamentaux en ont découlé. En fait, ces relations min-max sont, pour la plupart, des cas particuliers de la théorie de la dualité en programmation linéaire (voir section 12.2). Comme nous l'avons signalé plus haut, la dualité en programmation linéaire joue également un rôle fondamental dans la conception d'algorithmes d'approximation.

Que dire si G n'est plus biparti ? Dans ce cas, un couplage maximum peut être strictement plus petit que la couverture par sommets minimum. Par exemple, si G est un cycle de longueur impaire $2p+1$, la taille d'un couplage maximum est p, alors qu'une couverture par sommets optimale compte $p+1$ sommets. L'inégalité peut être stricte, même si le graphe admet un couplage parfait ; par exemple, le graphe de Petersen ci-dessous. Ce graphe admet un couplage parfait de taille 5 ; mais la taille d'une couverture par sommets minimum est 6. Ce graphe n'admet pas de couverture par sommets de taille 5 car toute couverture doit choisir au moins $p+1$ sommets de tout cycle de longueur impaire $2p+1$, pour couvrir les arêtes de ce cycle, et le graphe de Petersen a deux cycles disjoints de longueur 5.



Selon la conjecture **NP** $\neq$ co-**NP**, les problèmes **NP**-difficiles n'admettent pas de certificats négatifs. Ainsi, le problème de la couverture par sommets pour les graphes généraux, qui est **NP**-difficile, n'a-t-il pas de certificats négatifs, à moins que **NP** = co-**NP**. En revanche, le problème du couplage maximum pour les graphes généraux est dans **P**. Les certificats négatifs pour ce problème ne sont pas des couvertures par sommets, mais une structure plus générale : les couvertures impaires.

Une *couverture impaire* C d'un graphe $G = (V, E)$ est formée d'une suite de sous-ensembles $S_1, \ldots, S_k$ de cardinal impair de V et d'une suite $v_1, \ldots, v_\ell$ de sommets, telles que pour toute arête e de G, e a pour extrémité l'un des v_i, ou bien les deux extrémités de e appartiennent toute deux à l'un des S_j. Le *poids* de cette couverture C est défini par $w(C) = \ell + \sum_{i=1}^{k} (|S_i| - 1)/2$. Nous avons alors la relation min-max suivante :

Théorème 1.7 *Pour tout graphe,*

$$\max_{\textit{couplage } M} |M| = \min_{\textit{couverture impaire } C} w(C)$$

En général un couplage maximum peut être strictement plus petit qu'une couverture par sommets de taille minimum. Peut-il être arbitrairement plus petit ? La réponse est « Non ». Un corollaire du théorème 1.3 est que dans tout graphe la taille d'un couplage maximum est au moins la moitié de celle d'une couverture minimum. Plus précisément, le théorème 1.3 induit l'inégalité min-max suivante. Les algorithmes d'approximation conduisent fréquemment à ce genre d'inégalités min-max, toutes aussi intéressantes.

Corollaire 1.8 *Dans tout graphe,*

$$\max_{\textit{couplage } M} |M| \quad \leqslant \min_{\textit{couverture par sommets } U} |U| \quad \leqslant 2 \cdot \max_{\textit{couplage } M} |M|$$

Bien que le problème de la couverture par sommets n'admette pas de certificat négatif (à moins que **NP** = co-**NP**), il existe un moyen de certifier que (G, k) est une instance négative pour des valeurs suffisamment petites de k. L'algorithme 1.2 (plus précisément le minorant utilisé) apporte une réponse. Notons $\mathcal{A}(G)$ la taille de la couverture générée par l'algorithme 1.2. Alors, $\mathrm{OPT}(G) \leqslant \mathcal{A}(G) \leqslant 2 \cdot \mathrm{OPT}(G)$. Si $k < \mathcal{A}(G)/2$ alors $k < \mathrm{OPT}(G)$, et donc (G, k) est une instance négative. Ainsi, l'algorithme 1.2 produit un certificat négatif pour chaque instance (G, k) telle que $k < \mathrm{OPT}(G)/2$.

Un certificat négatif pour des instances (I, B) d'un problème de minimisation Π, telles que $B < \mathrm{OPT}(I)/\alpha$ est appelé un *certificat négatif approché de facteur* α. De même que pour les certificats positifs et négatifs habituels, ce certificat n'est pas nécessairement calculable en temps polynomial. Une α-approximation $\mathcal{A}$ pour Π produit un tel certificat, et puisque $\mathcal{A}$ est un algorithme polynomial, ce certificat est calculable en temps polynomial. Nous découvrirons au chapitre 27 un résultat fascinant : le problème du vecteur non nul le plus court d'un module admet un certificat négatif approché de facteur n ; mais on ne connaît aucun algorithme polynomial pour en construire.

1.3 Exercices

1.1 Proposez une 1/2-approximation pour le problème suivant :

Problème 1.9 (Sous-graphe sans cycle maximum) Étant donné un graphe orienté $G = (V, E)$, trouvez un sous-ensemble d'arêtes sans cycle et de taille maximum.
Indication : Numérotez arbitrairement les sommets de 1 à $|V|$ puis sélectionnez le plus grand des deux ensembles d'arêtes suivants : l'ensemble des arêtes croissantes (i, j) où $i < j$, et son complémentaire. Quelle méthode utilisez-vous pour majorer OPT ?

1.2 Proposez une 2-approximation pour trouver un couplage maximal de taille minimum dans un graphe non orienté.
Indication : Un couplage maximal est de taille au moins égale à la moitié de celle d'un couplage maximum.

1.3 (R. Bar-Yehuda) Voici une 2-approximation pour le problème de la couverture par sommets de taille minimum. Construisez un arbre de parcours en profondeur du graphe G, et indiquez l'ensemble S de tous les sommets internes de cet arbre. Démontrez que S couvre bien les arêtes de G et que $|S| \leqslant 2 \cdot \mathrm{OPT}$.
Indication : Démontrez que G possède un couplage de taille $\geqslant \lceil |S|/2 \rceil$.

1.4 Pour résoudre de façon approchée un problème d'optimisation, le premier réflexe est sans doute de proposer un algorithme glouton. Pour le problème de la couverture par sommets, un tel algorithme sélectionnerait un sommet de degré maximum, supprimerait ce sommet du graphe ainsi que ses arêtes incidentes et itérerait jusqu'à ce que le graphe n'ait plus d'arête. Démontrez que cet algorithme est une $O(\log n)$-approximation. Donnez une instance critique.
Indication : L'analyse est similaire à celle du théorème 2.4.

1.5 L'algorithme glouton suivant calcule un couplage maximal : sélectionner une arête, retirer ses deux extrémités du graphe et itérer jusqu'à ce que le graphe soit totalement déconnecté. Cela fait-il de l'algorithme 1.2 un algorithme glouton ?

1.6 Donnez un minorant pour la couverture par sommets de coût minimum.
Indication : Difficile sans utiliser la dualité en programmation linéaire.

1.7 Soit $A = \{a_1, \ldots, a_n\}$ un ensemble fini, et $\preccurlyeq$ un *ordre partiel* sur A, c'est-à-dire une relation réflexive, transitive et antisymétrique. Deux éléments a_i et a_j de A sont dits *comparables* si $a_i \preccurlyeq a_j$ ou $a_j \succcurlyeq a_i$, et *incomparables* sinon. Une *chaîne* est un sous-ensemble d'éléments de A deux à deux comparables. Une *antichaîne* est un sous-ensemble d'éléments de A deux à deux incomparables. Une *couverture par chaînes* (resp. *antichaînes*) de A est une partition de A en chaînes (resp. antichaînes). La taille d'une telle couverture est le nombre de chaînes (resp. antichaînes) qui la composent. Démontrez la relation min-max suivante : la taille (longueur) de la chaîne la plus longue est égale à la taille de la plus petite couverture par antichaînes.
Indication : Soit m la longueur de la chaîne la plus longue. Soit $\phi(a)$ la longueur de la chaîne la plus longue issue de a, c'est-à-dire constituée d'éléments plus grands que l'élément a. Considérez la partition de A par les $A_i = \{a \in A \mid \phi(a) = i\}$, pour $1 \leqslant i \leqslant m$.

1.8 (Théorème de Dilworth, voir [203]) Montrez que pour tout ordre partiel fini, la longueur de l'antichaîne maximum est égale à la taille minimum d'une couverture par chaînes.
Indication : Appliquez le théorème de König et Egerváry. Pour un ensemble partiellement ordonné A de taille n, considérez le graphe biparti $G = (U, V, E)$, où $|U| = |V| = n$ et $(u_i, v_j) \in E$ ssi $i \neq j$ et $a_i \preccurlyeq a_j$.

Les dix exercices suivants utilisent des notions abordées dans l'annexe A.

1.9 Le problème d'optimisation suivant est-il dans **NP** ? Soient $G = (V, E)$ un graphe non orienté muni d'une fonction de coût $c : V \to \mathbf{Q}^+$ sur les sommets et k un entier positif, trouver une couverture par sommets de coût minimal ayant au plus k sommets.
Indication : Peut-on décider de la validité d'une instance (c'est-à-dire si une instance admet au moins une solution réalisable) en temps polynomial ?

1.10 Soient Π un problème de minimisation **NP** et $\mathcal{A}$ un algorithme d'approximation randomisé pour Π, tel que pour toute instance, l'espérance du coût de la solution produite est $\leqslant \alpha \cdot \mathrm{OPT}$, avec $\alpha > 1$. Quel est le meilleur facteur d'approximation que l'on puisse garantir pour Π en utilisant $\mathcal{A}$?
Indication : Une garantie de $2\alpha - 1$ s'obtient facilement. Pour approcher arbitrairement le facteur α, exécutez l'algorithme un nombre polynomial de fois et sélectionnez la meilleure solution. Appliquez la borne de Chernoff.

1.11 Démontrez que si SAT est **NP**-difficile, et que s'il existe une réduction polynomiale de SAT au problème de la couverture par sommets de taille minimale, alors ce dernier est aussi **NP**-difficile.
Indication : Montrez que la composition de deux réductions polynomiales est également une réduction polynomiale.

1.12 Démontrez que si le problème de la couverture par sommets de taille minimale est dans co-**NP**, alors **NP** = co-**NP**.

1.13 (Pratt [231]) Démontrez que le langage des nombres premiers est dans **NP** ∩ co-**NP**.
Indication : Considérez le groupe multiplicatif des éléments inversibles modulo n, $Z_n^* = \{a \in Z_n \mid \text{pgcd}(a, n) = 1\}$. Clairement, $|Z_n^*| \leqslant n - 1$. Utilisez le fait que $|Z_n^*| = n - 1$ ssi n est premier, et que Z_n^* est cyclique si n est premier. Un certificat positif consiste en un générateur de Z_n^*, une factorisation de $n-1$ et récursivement, ces mêmes informations pour chaque facteur premier de $n-1$. Attention, vos algorithmes doivent être polynomiaux en $\log(n)$, la taille de n.

1.14 Prouvez que les problèmes d'optimisation présentés dans ce livre, sont autoréductibles : en particulier, le couplage maximum, MAX-SAT (problème 16.1), la clique maximum (problème 29.15), le surfacteur minimum (problème 2.9), et l'ordonnancement de temps d'exécution minimum (problème 10.1).
Indication : Pour la clique maximum, envisagez les deux possibilités : v est ou n'est pas dans la clique maximum ; et restreignez G à v et ses voisins ou retirez v de G, suivant le cas. Pour la surchaîne minimale, éliminez deux sous-chaînes et remplacez-les par une surchaîne valide (éventuellement la simple concaténation). Si la longueur de la surchaîne minimale est inchangée, alors continuez avec l'instance la plus courte. Généralisez légèrement le problème d'ordonnancement — considérez que l'instance du problème contient également le nombre d'unités de temps déjà ordonnancées sur chaque machine.

1.15 Proposez une définition d'autoréductibilité pour les problèmes **NP**, c'est-à-dire pour les problèmes de décision et non les problèmes d'optimisation, qui permette de trouver en temps polynomial une solution réalisable à partir d'un oracle pour le problème de décision associé, et qui, en sus, génère un arbre d'autoréductibilité pour chaque instance.
Indication : Ordonnez arbitrairement les « atomes »d'une solution, par exemple, pour SAT, ordonnez arbitrairement les n variables.

1.16 Soient Π_1 et Π_2 deux problèmes d'optimisation tels qu'il existe une réduction isofacteur[7] de Π_1 à Π_2. Montrez que s'il existe une α-approximation pour Π_2, il en existe une pour Π_1.

[7] *Approximation factor preserving reduction*, en anglais.

Indication : Commencez par montrer que si la réduction transforme une instance I_1 de Π_1 en une instance I_2 de Π_2 alors $\mathrm{OPT}_{\Pi_1}(I_1) = \mathrm{OPT}_{\Pi_2}(I_2)$.

1.17 Montrez que $L \in \mathbf{ZPP}$ ssi $L \in (\mathbf{RP} \cap \text{co-}\mathbf{RP})$.

1.18 Montrez que si $\mathbf{NP} \subseteq \text{co-}\mathbf{RP}$ alors $\mathbf{NP} \subseteq \mathbf{ZPP}$.
Indication : Si une instance ϕ de SAT est satisfaisable, alors on peut calculer avec forte probabilité une instanciation positive des variables en utilisant l'autoréductibilité et la machine co-**RP** pour SAT. Si ϕ n'est pas satisfaisable, la machine co-**RP** le confirme par une réponse négative, avec forte probabilité.

1.4 Notes

La notion de problème bien caractérisé a été introduite par Edmonds dans [76], puis a été formalisée précisément par Cook [54]. Dans le même article, Cook initia la théorie de la **NP**-complétude, découverte indépendamment par Levin [194]. Ces travaux prirent leur véritable valeur avec les travaux de Karp [172], démontrant que des problèmes algorithmiques fondamentaux très différents sont tous **NP**-complets.

Remarquons que les premiers algorithmes d'approximation ont été conçus avant la théorie de la **NP**-complétude : par Vizing [264] pour la coloration d'arêtes minimale, par Graham [120] (problème 10.1) pour minimiser le temps d'exécution d'un ordonnancement et par Erdös [80] pour MAX-CUT (problème 2.14). L'importance de concevoir de tels algorithmes s'est imposée à mesure que la conjecture $\mathbf{P} \neq \mathbf{NP}$ devenait plus vraisemblable. La définition d'algorithmes d'approximation a été posée formellement par Garey, Graham et Ullman [98] et Johnson [158]. Le premier algorithme à base de programmation linéaire a été conçu par Lovász [200] afin d'analyser l'algorithme glouton pour le problème de la couverture par ensembles (voir chapitre 13). Rabin [233], quant à lui, mena les tout premiers travaux exploratoires sur l'utilisation de la randomisation en algorithmique — cette notion est particulièrement utile en algorithmique d'approximation. Le théorème 1.7 fut prouvé par Edmonds [76] et l'algorithme 1.2 conçu indépendamment par Gavril et Yannakakis (voir [100]).

Nous conseillons la lecture des livres de Cormen, Leiserson, Rivest, et Stein [57], Papadimitriou et Steiglitz [226], et Tarpan [255] pour les techniques algorithmiques classiques. Le livre de Lovász et Plummer [203] propose un bon traitement des relations min-max. Pour les algorithmes d'approximation : Hochbaum [134] et Ausiello, Crescenzi, Gambosi, Kann, Marchetti, et Protasi [19]. Référez-vous aux sections 12.5, A.6, et B.4 pour la programmation linéaire, la théorie de la complexité, et les algorithmes randomisés, respectivement.