Algorithmes Randomisés

Clément Legrand
TIPE ENS, 2016/2017

1 Introduction

Généralités sur les algorithmes randomisés et d'approximation

Les algorithmes randomisés sont des algorithmes utilisant une suite de bits "aléatoires" en plus de l'instance donnée en entrée pour résoudre un problème [6]. Il est toutefois important de noter que l'exécution de l'algorithme ne dépend que de la suite de bits aléatoires et de l'instance : si on lui soumet deux fois les mêmes entrées, il s'exécutera de la même manière. On distingue deux grandes familles d'algorithmes randomisés : les algorithmes dits de *Monte Carlo* et ceux de *Las Vegas*. Les premiers fournissent une solution approchée au problème en un temps polynomial en la taille de l'instance et indépendant de la suite de bits aléatoire utilisée. Ils sont dits efficaces s'il est possible de borner l'espérance de l'éloignement de leurs solutions par rapport à la solution optimale. Ceux dits de Las Vegas renvoient une solution exacte en un temps dépendant de la suite aléatoire et pouvant être arbitrairement grand mais dont l'espérance est polynomiale en la taille de l'instance [6, 5]. Nous nous intéresserons dans ce TIPE uniquement aux algorithmes dits de Monte Carlo. On dit d'un algorithme d'approximation randomisé A que c'est une k-approximation pour le problème de maximisation $\mathcal{P}$ si $k = \inf_{I \in \mathcal{P}} \frac{A(I)}{Opt(I)}$ est non nul. k est alors appelé facteur d'approximation et plus il est élevé et proche de 1, plus on considère que l'algorithme est bon.

Le recours au hasard présente de nombreux avantages: il permet par exemple souvent d'éviter des choix pathologiques systématiques, ou de contrer un adversaire "malicieux" [5]. Ces algorithmes ont aussi souvent des performances bien meilleures que des algorithmes déterministes en terme de temps d'exécution comme en terme de qualité. Par ailleurs, la notion de complexité moyenne ou d'espérance du résultat est relativement récente en théorie de la complexité et permet des analyses plus fines de la complexité réelle de certains problèmes [1]. Enfin, ils s'imposent parfois comme seules solutions raisonnables pour un problème donné.

Problème de la coupe maximale

Soit $G = (V, E)$ un graphe non orienté constitué d'un ensemble de sommets V et d'un ensemble d'arêtes E. On munit E d'une fonction de poids w. Une coupe est une partition $C = \{A, B\}$ de V et le poids d'une coupe C est la somme des poids des arêtes ayant une extrémité dans A et une autre dans B. Nous ne considérerons par la suite que des graphes dont les poids des arêtes sont positifs ou nuls. Cette hypothèse nous permet d'affirmer que la somme totale des poids des arêtes est supérieure au poids de la coupe maximale.

Ce problème a des applications en physique statistique à travers le Modèle d'Ising, consistant à trouver quel spin attribuer à chaque atome d'un système afin de minimiser son énergie totale, ainsi que dans le Problème de Pinter (perçage optimal d'une plaque pour un circuit imprimé sur les deux faces).

Ce problème est dit NP-complet: on ne connaît pas d'algorithme calculant la solution optimale en temps polynomial à ce jour. De plus, ce problème est relativement difficilement approximable: il n'existe pas de PTAS dans le cas général (un PTAS est un algorithme calculant, pour tout $\epsilon > 0$ donné, une solution proche à un facteur $1-\epsilon$ de l'optimal, et ce en temps polynomial mais dépendant de $1/\epsilon$ bien sur). En revanche, il existe des algorithmes polynomiaux dans le cas des graphes planaires et un PTAS dans celui des graphes denses. Nous étudierons ici des algorithmes d'approximation et de Monte Carlo.

Un premier algorithme randomisé extrêmement simple a été présenté par Sahni et Gonzales en 1976 : répartir chacun des sommets entre A et B avec une probabilité $1/2$. Cet algorithme est une $1/2$-approximation et peut être dérandomisé en un algorithme glouton de même facteur d'approximation [6]. D'autres facteurs d'approximation légèrement supérieurs ont été par la suite trouvés, mais aucun ne constituait en une avancée révolutionnaire, jusqu'à ce que Goemans et Williamson proposent en 1995 un algorithme plus complexe, de facteur d'approximation environ 0,878 [3, 4]. Il est possible, sous certaines hypothèses, de montrer que ce facteur d'optimisation est optimal.

2 Algorithmes

2.1 Algorithme glouton

Principe On construit la coupe (A, B) progressivement, en plaçant chaque sommet de manière à maximiser le poids de la coupe déjà construite. Concrètement, supposons deux ensembles A et B déjà construits. Soit $u \in V \backslash (A \cup B)$. Pour toute partie $X \subset V$, on notera $w_{uX} := \sum_{x \in X} w_{ux}$. On a: poids de la coupe$(A \cup \{u\}, B) =$ poids de la coupe$(A, B) + w_{uB}$. L'algorithme ajoute donc u à A si $w_{uB} \geqslant w_{uA}$ et inversement. On notera que l'ordre dans lequel on examine les éléments de V influe sur la solution proposée par l'algorithme. Ainsi il est ici possible de tirer au hasard une permutation de l'ordre dans lequel l'algorithme examinera le sommets. À chaque étape, il n'est pas nécessaire de recalculer entièrement les poids des coupes construites car celles-ci sont presque identiques et cet algorithme s'exécute donc en temps quadratique en $|V|$.

1/2 approximation D'après le calcul précédent, le poids de la coupe construite augmente d'au moins $\frac{1}{2}(w_{uA} + w_{uB})$ à chaque itération. En définitive, chaque arête aura été traitée une et une seule fois. Par conséquent, le poids de la coupe retournée est supérieur à la moitié de la somme totale des poids des arêtes du graphe. L'algorithme est donc une $1/2$ approximation, et ce quel que soit l'ordre dans lequel on examine les éléments de V.

2.2 Premier algorithme randomisé

Principe Pour chaque somment, on tire à pile ou face s'il va dans A ou dans B. On obtient ainsi une partition aléatoire telle que chacune des partitions possibles de V est équiprobable et obtenue avec une probabilité $1/2^n$. Il faut pour calculer le poids de cette coupe sommer les poids d'au plus $|E|$ arêtes. Cet algorithme s'exécute donc également en temps quadratique en $|V|$.

1/2-approximation Chaque sommet ayant une chance sur deux d'appartenir à A, chaque arête a une chance sur deux de contribuer au poids de la coupe. L'espérance du poids de la coupe obtenue vaut donc la moitié de la somme totale des poids des arêtes, somme supérieure au poids de la coupe maximale. Cet algorithme est donc une $1/2$ approximation (cette borne est atteinte exactement sur les graphes bipartis).

Variance et répétitions Chacune des partitions (A, B) de V étant équiprobables, on s'attend à ce que la variance du poids de la solution retournée soit assez élevée. Le poids des solutions retournées variant beaucoup, ceci nous amène intuitivement à penser qu'il est possible d'améliorer considérablement l'espérance de la solution en exécutant l'algorithme quelques fois avant de prendre le maximum des solutions trouvées. La loi du maximum du poids de la coupe retournée sur plusieurs tirages étant inconnue, on essayera de déterminer empiriquement le nombre de tirages nécessaires pour obtenir une coupe de poids de "bonne qualité".

2.3 Algorithme dérandomisé

Principe et algorithme Supposons qu'à toute itération de l'exécution de l'algorithme randomisé, on sache calculer (en temps polynomial) l'espérance conditionnelle du poids de la coupe, sachant la répartition déjà effectué des précédents sommets, en fonction du choix de la répartition du sommet courant. Il est alors possible de construire un algorithme dérandomisé répartissant chaque sommet de manière a maximiser l'espérance conditionnelle de la coupe.

1/2 approximation À chaque itération l'espérance de la coupe finale sachat la coupe en partie construite ne peut qu'augmenter. Au départ, cette espérance est celle de l'algorithme randomisé. Cet algorithme est donc une $1/2$-approximation.

Montrons qu'en réalité, cet algorithme s'exécute de manière identique à l'algorithme glouton. Notons $\mathbb{E}(\text{poids de la coupe}|(C1, C2))$ l'espérance du poids de la coupe sachant

$(C1, C2)$ déjà construit. Supposons (A, B) déjà construit. Soit $u \in V \backslash (A \cup B)$.

$$\begin{aligned}
\mathbb{E}(\text{poids de la coupe}|A \cup \{u\}, B) &= \text{poids de la coupe}(A \cup \{u\}, B) + \frac{1}{2} \sum_{x \in V \backslash (A \cup B \cup \{u\})} w_{xV} \\
&= \text{poids de la coupe}(A, B) + w_{uB} + \frac{1}{2} \sum_{x \in V \backslash (A \cup B \cup \{u\})} w_{xV} \\
&= \mathbb{E}(\text{poids de la coupe}|A, B \cup \{u\}) + w_{uB} - w_{uA}
\end{aligned}$$

Ainsi u est ajouté à A si $w_{uB} \geqslant w_{uA}$, cet algorithme s'exécute donc de la même manière que l'algorithme glouton vu précédemment. On retiendra cependant le principe de la dérandomisation, principe applicable à de nombreux algorithmes randomisés à condition que l'espérance conditionnelle puisse à tout moment se calculer en temps raisonnable.

2.4 Algorithme de Goemans

M.Goemans et D.Williamson ont proposé en 1995 [3, 4] un algorithme randomisé pour le problème de la coupe maximale de facteur d'approximation $\frac{2}{\pi} \inf_{\theta \in]0,\pi[} \frac{\theta}{1-\cos\theta} \approx 0,878567$. Nous ne l'implémenterons pas car le problème de programmation semi-définie positive auquel on se ramène est difficile à mettre en œuvre. Je me suis en revanche penché sur la partie de la preuve concernant directement le problème de la coupe maximale, qui est assez éclairante est mais est trop longue pour être présentée dans ce rapport.

3 Analyse expérimentale

L'algorithme randomisé et l'algorithme glouton vus précédemment ont les mêmes facteurs d'approximation et les mêmes complexités théoriques. Mais ces facteurs d'approximations correspondent à des pires cas sur l'ensemble des instances, qui surviennent peut être très rarement en "réalité". D'autre part, l'écart entre l'espérance de la solution retournée et la moitié de l'optimal est a priori souvent élevé. La Figure 1 illustre ce problème en représentant la distribution des poids de toutes les coupes possibles d'un graphe donné. Le positionnement de l'espérance de la solution retournée par rapport à la courbe de répartition des poids pour ce graphe nous conforte dans l'idée qu'effectuer des répétitions améliorera notablement l'algorithme et permettra peut être de rattraper l'algorithme glouton.

Il est donc intéressant de voir ce qu'il en est en pratique et nous étudions donc ce problème pour des graphes uniformes (modèle Erdös-Renyi). La solution optimale n'étant pas calculable pour de grand graphes, nous faisons systématiquement deux études: l'une sur le facteur d'approximation $k(G) = \frac{\mathbb{E}(A(G))}{Opt(G)}$ pour des graphes G de petites tailles ($|V| \leqslant 15$) et une sur le poids des coupes pour des graphes plus gros ($|V| \in \{100, 200, 1\,000\}$).

3.1 Influence de la taille du graphe

En énumérant toutes les coupes, il est déraisonnable d'espérer trouver une solution exacte en temps raisonnable pour des graphes de plus de vingts sommets. On retrouve bien les complexité temporelles attendues, à savoir exponentielle pour l'algorithme optimal (voir Figure 2(a)) et quadratique pour l'algorithme randomisé et pour l'algorithme glouton (voir Figure 2(b)). On peut remarquer que l'algorithme randomisé s'exécute environ deux fois plus rapidement que l'algorithme glouton.

En terme de qualité de la solution, on remarque que quand le graphe est petit, l'algorithme randomisé est peu performant (voir Figure 3(a)). D'autre part la variance du poids de la coupe étant élevée par rapport au poids de la coupe maximale quand le nombre d'arête est faible, il est peu fiable. Le facteur de qualité semble augmenter avec la taille du graphe. Ce rapport n'est pas calculable quand le graphe est trop grand, on observe néanmoins que l'écart relatif entre les solutions fournies par l'algorithme randomisé et le glouton tend vers 0 quand la taille augmente (voir Figure 3(b)). L'écart entre les facteurs de qualité tend donc lui aussi vers 0.

3.2 Influence de la densité du graphe

Encore une fois, pour des densités faibles, le nombre d'arêtes étant faible, l'algorithme randomisé est peu performant (voir Figure 4(a)), et l'écart relatif des coupes retournées par les deux algorithmes d'approximation est plus faible pour des graphes plus grands et plus denses (voir Figure 4(b)).

3.3 Influence du nombre de répétitions

Il apparaît qu'effectuer quelques répétitions permet d'augmenter notablement la qualité de la solution fournie par l'algorithme randomisé (voir Figure 5(a) et Figure 5(b)). En revanche, cela ne permet pas, même pour un nombre élevé de répétitions, de rattraper l'algorithme glouton (voir Figure 5(b)). Cela est très probablement dû au fait que pour ce type de graphe, la proportion des coupes de poids supérieurs à celui retourné par l'algorithme glouton reste assez petite (voir Figure 1).

4 Conclusion

Si ces algorithmes sont des $1/2-$approximations, il apparaît tout de même qu'en pratique leur facteur de qualité pour des graphes de types Erdös-Renyi est bien supérieur à $1/2$. On retiendra aussi le fait que l'algorithme randomisé de base est finalement peu fiable et peu efficace quand le nombre d'arêtes est faible (densité faible ou petit graphe). L'idée d'effectuer des répétitions pour améliorer l'algorithme randomisé ne fonctionne que dans une certaine mesure: certes cela permet d'améliorer significativement la qualité de la coupe, mais cela ne suffit pas pour rivaliser avec l'algorithme glouton, même pour un assez grand nombre de répétitions.

A Annexe

A.1 Bibliographie

Ne sont cités ici que les ouvrages et sites dont je me suis le plus servi.

[1] Vidéo cours youtube mpri et ens schabanel. http://youtube.com.

[2] T. H. CORMEN, C. E. LEISERSON et R. L. RIVEST : *Introduction à l'algorithmique*. Dunod, 2002.

[3] M. X. GOEMANS et D. P. WILLIAMSON : Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, (6):1115–1145, nov 1995. http://www-math.mit.edu/~goemans/PAPERS/maxcut-jacm.pdf.

[4] F. MEUNIER et A. SEBŐ : *Parcours et coupes*, chap. 13. Hermes Édition, 2007. http://cermics.enpc.fr/~meuniefr/hermes13dec.pdf.

[5] R. MOTWANI et PRABHAKAR : *Randomized Algorithms*. Cambridge University Press, 1995.

[6] N. SCHABANEL : Algorithmes d'approximation et algorithmes randomisés. http://pauillac.inria.fr/~quercia/documents-info/Luminy-2003/schabanel/schabanel.pdf, may 2003. Cours donné lors du Stage de Luminy 2003 à destination des enseignants d'informatique de CPGE.

A.2 Figures

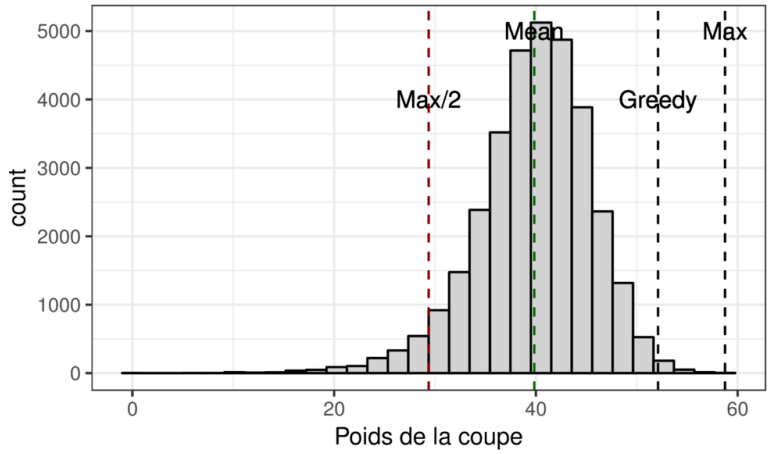
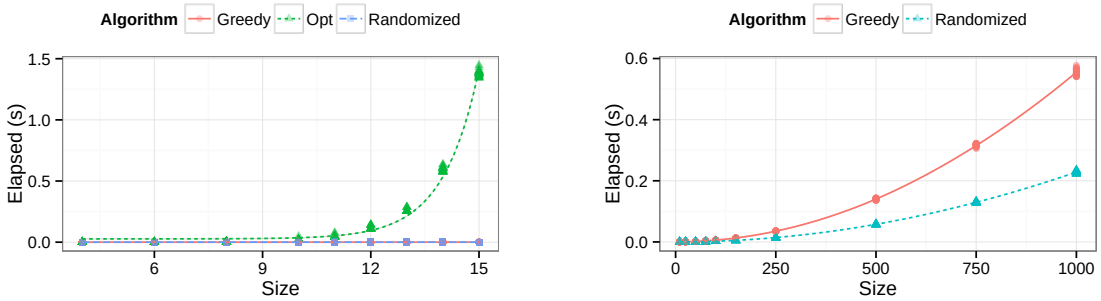


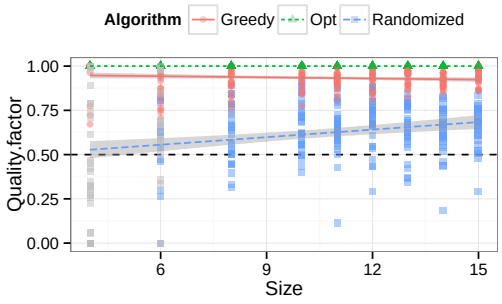
Figure 1: Exemple de répartition des poids des coupes retournées par l'algorithme randomisé (pour un graphe $|V| = 15$)



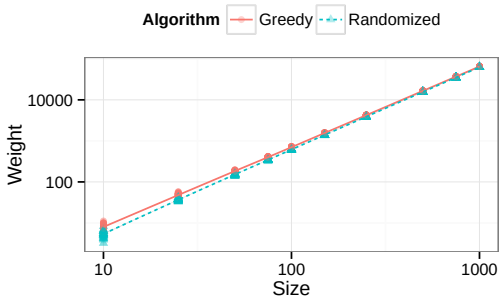
(a) Temps d'exécution.

(b) Temps d'exécution des algorithmes d'approximation.

Figure 2: Influence de la taille du graphe sur le temps d'exécution des différents algorithmes.

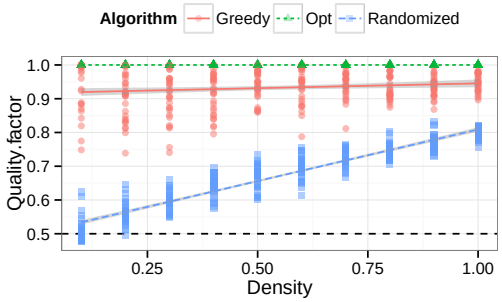


(a) Influence sur le facteur de qualité.

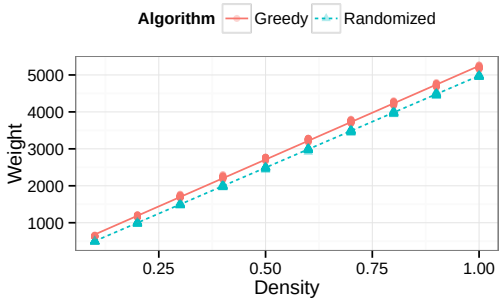


(b) Influence sur le poids de la coupe.

Figure 3: Influence de la taille du graphe sur la qualité des solutions des différents algorithmes.

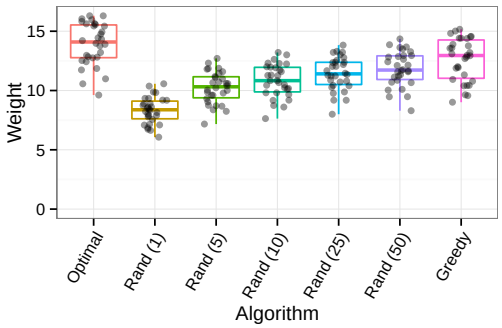


(a) Influence sur le facteur de qualité $|V| = 15$.

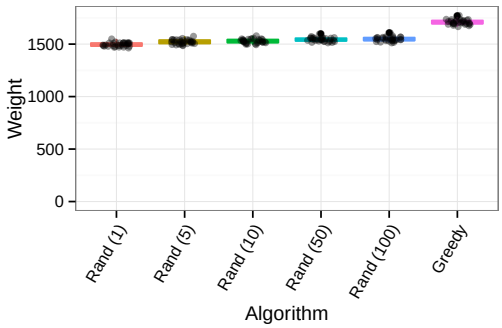


(b) Influence sur le poids de la coupe $|V| = 200$.

Figure 4: Influence de la densité du graphe sur la qualité des solutions des différents algorithmes.



(a) Influence sur le facteur de qualité $|V| = 15$.



(b) Influence sur le poids de la coupe $|V| = 200$.

Figure 5: Influence du nombre de répétitions sur la qualité des solutions des différents algorithmes.