IFT800 - Série d'exercices #3

Manuel Lafond

Algorithmes probabilistes

Exercice 1. Considérez le problème MAX-k-SAT dans lequel chaque clause a k variable ou plus. Quel ratio d'approximation probabiliste pouvez-vous atteindre? Et quel ratio d'approximation déterministe pouvez-vous atteindre?

Solution.

Supposons qu'on choisit une affectation des variables x_i au hasard, avec $x_i = true$ et $x_i = false$ avec probabilité 1/2 chacune. Considérons une clause C_j et supposons qu'elle a $h \geq k$ variables. Il n'y a qu'une seule façon de ne pas satisfaire C_j, et ceci survient avec probabilité $1/2^h$. On va donc satisfaire C_j avec probabilité $1 - 1/2^h \geq 1 - 1/2^k$. Si I_{C_j} est une variable indicatrice avec $I_{C_j} = 1$ si C_j est satisfaite et $I_{C_j} = 0$ sinon, on peut calculer l'espérance de APP avec

$$\mathbb{E}[APP] = \mathbb{E}[\sum_{j=1}^{m} I_{C_j}] = \sum_{j=1}^{m} \mathbb{E}[I_{C_j}] \geq \sum_{j=1}^{m} (1 - 1/2^k) = m(1 - 1/2^k)$$

Puisque $OPT \leq m$, on a une $(1 - 1/2^k)$-approximation probabiliste.

On peut atteindre ce ratio de façon déterministe avec exactement le même algorithme de dérandomisation que celui présenté en classe.

Exercice 2. Considérez le problème MAX-k-CUT, où il faut séparer V en k ensembles et maximiser le nombre d'arêtes traversantes. Quel ratio d'approximation probabiliste pouvez-vous atteindre?

Solution.

Supposons qu'on démarre avec les ensembles $V_1, V_2, \ldots, V_k$ vides et que, pour chaque $u \in V$, on choisit au hasard l'emplacement de u parmi ces ensembles de façon uniforme.

Soit $uv \in E$. Il y a k choix pour l'emplacement de u et k choix pour l'emplacement de v. Il y a donc k^2 combinaisons de choix possibles. Dans k de ces choix, u et v sont dans la même partie et l'arête ne traverse pas. On a donc $k/k^2 = 1/k$ chances que uv ne soit pas traversante, et donc $1 - 1/k$ chances qu'elle traverse.

Si I_{uv} est une variable indicatrice avec $I_{uv} = 1$ si uv est traversante et $I_{uv} = 0$ sinon, on peut calculer l'espérance de APP avec

$$\mathbb{E}[APP] = \mathbb{E}[\sum_{j=1}^{m} I_{uv}] = \sum_{j=1}^{m} \mathbb{E}[I_{uv}] = \sum_{j=1}^{m} (1 - 1/k) = m(1 - 1/k)$$

Puisque $OPT \leq m$, on a une $(1 - 1/k)$-approximation probabiliste.

Exercice 3. Considérez l'algorithme probabiliste suivant pour VERTEX-COVER. On construit notre ensemble X itérativement. Pour chaque arête $uv \in E$, on inclut u dans X avec probabilité $1/2$ s'il n'y est pas déjà, ou bien v dans X avec probabilité $1/2$ s'il n'y est pas déjà.

Il n'est pas évident d'analyser $\mathbb{E}[APP]$ par rapport à OPT.

On va simplifier et supposer que dans G, chaque sommet a exactement 2 voisins. Dans ce cas, que vaut $\mathbb{E}[APP]$? Et quel est le ratio d'approximation probabiliste? Ensuite, posez les même questions, mais généralisé au cas où chaque sommet a exactement k voisins.

Solution.

Pour calculer $\mathbb{E}[APP]$, on va calculer la probabilité qu'un sommet $u \in V$ soit inclut dans X. On nous dit que u fait partie de deux arêtes, disons uv et uw. Il y a 4 combinaisons d'inclusion possibles pour l'algorithme lorsque uv et uw sont considérés : on inclut (1) v et w; (2) v et u; (3) u et w; (4) u et u (dans ce dernier cas, on n'ajoute u qu'une seule fois, mais c'est une possibilité). Un seul de ces choix n'inclut pas u, ce qui survient avec probabilité $1/4$. Donc on va inclure u avec probabilité $1 - 1/4 = 3/4$.

Si I_u est une variable indicatrice avec $I_u = 1$ si u est inclut dans X et $I_v = 0$ sinon, on peut calculer l'espérance de APP avec

$$\mathbb{E}[APP] = \mathbb{E}[\sum_{u \in V} I_u] = \sum_{u \in V} \mathbb{E}[I_u] = \sum_{u \in V} 3/4 = 3n/4$$

Puisque chaque sommet couvre au plus 2 arêtes, toute couverture aura au moins $|E|/2$ sommets. Donc $OPT \geq |E|/2$. De plus, puisque chaque sommet a exactement 2 voisins, $|E| = 2n/2 = n$, c'est-à-dire $OPT \geq n/2$. Puisque $\mathbb{E}[APP] = 3n/4$, pour calculer le ratio d'approximation, on cherche c tel que

$$3n/4 = c \cdot n/2$$

et on trouve $c = 3/2$. On a donc une 3/2-approximation probabiliste.

Notez que ce n'est pas un grand accomplissement, car il est trivial de trouver une couverture par sommet optimale si chaque sommet a 2 voisins.

Pour généraliser à k voisins, on note que si u est dans k arêtes, il y a 2^k combinaisons de choix effectué par ces arêtes, et seulement une d'entre elle ne choisit pas u. Donc on a seulement $1/2^k$ chances de ne pas choisir u et $1 - 1/2^k$ chances de l'inclure. L'espérance devient $\mathbb{E}[APP] = n(1 - 1/2^k)$. On peut argumenter que $OPT \geq |E|/k$. De plus, $|E| = kn/2$ car chaque sommet a k voisins et on déduit que

$$OPT \geq |E|/k = kn/(2k) = n/2$$

Notez que ceci fonctionne seulement car *chaque* sommet a k voisins. Le ratio d'approximation c est trouvé en isolant dans

$$n(1 - 1/2^k) = c \cdot n/2$$

et on trouve $c = 2 \cdot (1 - 1/2^k)$.

LP et ILP

Exercice 4. Cet exercice sert à entraîner nos conversions problème-ILP. Le but n'est pas de développer un algorithme d'approximation.

1. Exprimez le problème de trouver un matching de poids maximum avec un ILP (le poids d'un matching est la somme des poids des arêtes dans le matching).

2. Exprimez le problème du sac à dos avec un ILP (voir notes de cours, le sac à dos est dans la table des matières).

3. Revenons au problème de supprimer tous les triangles d'un graphe en supprimant un minimum d'arêtes. Exprimez ce problème avec un ILP.

4. Dans le problème CLUSTER-DELETION, on reçoit un graphe G et on doit supprimer un minimum d'arêtes afin que chaque composante connexe de G soit une clique. Exprimez ce problème avec un ILP.

   Notez que dans un graphe, chaque composante connexe est une clique si et seulement si $\forall u, v, w$, l'affirmation $uv \in E \wedge vw \in E \Rightarrow uw \in E$ est vraie. Vous pouvez utiliser ce fait en boîte noire, mais rien ne vous empêche de le démontrer pour vos apprentissages.

Solution.

MATCHING MAXIMUM

Une variable x_{uv} représente "mettre uv dans le matching". Il suffit d'éviter de mettre deux arêtes qui partagent un sommet.

Maximiser

$$\sum_{uv \in E} w_{uv} x_{uv}$$

Sujet à

$$x_{uv} + x_{yz} \leq 1 \quad \text{pour tout } uv, xy \in E \text{ tels que } \{u, v\} \cap \{x, y\} \neq \emptyset$$
$$x_{uv} \in \{0, 1\} \qquad \text{pour tout } uv \in E$$

SAC-À-DOS

Une variable x_i représente "mettre l'objet i dans le sac". Il suffit de ne pas dépasser la capacité.

Maximiser

$$\sum_{i=1}^{n} v_i x_i$$

Sujet à

$$\sum_{i=1}^{n} w_i x_i \leq W$$
$$x_i \in \{0, 1\} \qquad \text{pour tout } i = 1..n$$

SUPPRESSION DE TRIANGLES

Ici, x_{uv} représente "supprimer l'arête uv".

Minimiser

$$\sum_{uv \in E}^{n} x_{uv}$$

Sujet à

$$x_{uv} + x_{vw} + x_{uw} \geq 1 \qquad \text{pour tout triangle } u, v, w \text{ dans } G$$
$$x_{uv} \in \{0, 1\} \qquad \text{pour tout } uv \in E$$

CLUSTER-DELETION

Ici, x_{uv} représentera "l'arête uv n'est **pas** présente". Ceci est parfois confondant, car $x_{uv} = 1$ veut dire que uv est absente. Ceci donne toutefois le ILP le plus simple. On va créer la variable x_{uv} même quand $uv \notin E$. Dans ce cas, il faudra forcer $x_{uv} = 1$.

Il suffit de s'assurer de faire respecter la condition $uv \in E \wedge vw \in E \Rightarrow uw \in E$. Ceci est équivalent à $\neg(uv \in E \wedge vw \in E) \vee uw$ (le $\neg$ veut dire "négation"). Ceci est ensuite équivalent à

$$\neg(uv \in E) \vee \neg(vw \in E) \vee uw \in E$$

Puisque $x_{uv} = 1$ si $uv \notin E$, pour exprimer $uv \in E$, il faut écrire $1 - x_{uv}$. On peut donc exprimer l'expression ci-haut avec

$$1 - (1 - x_{uv}) + 1 - (1 - x_{vw}) + 1 - x_{uw} \geq 1$$

ce qui revient à

$$x_{uv} + x_{vw} - x_{uw} \geq 0$$

Ce qui est logique : si on laisse uv, vw mais que uw n'est pas présent, $x_{uv} + x_{vw} - x_{uw} = 0 + 0 - 1 < 0$.

Le ILP devient

Minimiser

$$\sum_{uv \in E}^{n} x_{uv}$$

Sujet à

$x_{uv} + x_{vw} - x_{uw} \geq 0$	pour tout triplet u, v, w de V
$x_{uv} \in \{0, 1\}$	pour tout $uv \in E$
$x_{uv} = 1$	pour tout $uv \notin E$

Exercice 5. Considérez la version avec poids de VERTEX-COVER. C'est-à-dire, chaque sommet v a un poids $f(v)$ et on cherche un ensemble $X \subseteq V$ qui couvre chaque arête et qui minimise $\sum_{x \in X} f(x)$. Donnez une 2-approximation pour ce problème.

Je recommande de formuler un LP et d'arrondir comme en cours.

Solution.

Pour les fins du LP, on va écrire $f(v) = w_v$. Notez que w_v n'est pas une variable, car sa valeur est connue. On peut donc utiliser w_v librement dans notre LP. Pour chaque $v \in V$, on aura une varible x_v représentant "v est dans notre couverture".

$$\begin{aligned} &\text{Minimiser} \\ &\qquad \sum_{v \in V}^{n} w_v x_v \\ &\text{Sujet à} \\ &\qquad x_u + x_v \geq 1 \qquad \text{pour tout } uv \in E \\ &\qquad 0 \leq x_v \leq 1 \qquad \text{pour tout } v \in E \end{aligned}$$

Considérez le même algorithme que celui vu en classe. C'est-à-dire, on résout le LP et on obtient un vecteur $\boldsymbol{x}^*$ optimal pour le LP. L'algorithme retourne $\{v \in V : x_v^* \geq 1/2\}$.

On va argumenter que c'est une 2-approximation. Notons d'abord que $OPT \geq OPT_{LP}$. Pour APP, soit X l'ensemble retourné par notre algorithme. On peut réutiliser les arguments vus en classe pour arguementer que X est une solution faisable (couvre chaque arête). Pour chaque $v \in X$, il faut que $x_v^* \geq 1/2$. Ceci revient à dire que $2x_v^* \geq 1$ pour chaque $v \in X$. On a donc

$$APP = \sum_{v \in X} w_v = \sum_{v \in X} w_v \cdot 1 \leq \sum_{v \in X} w_v \cdot 2x_v^* = 2 \sum_{v \in X} w_v x_v^* \leq 2 \sum_{v \in V} w_v x_v^* = 2 \cdot OPT_{LP}$$

et on a donc une 2-approximation.

Exercice 6. Considérez le LP pour VERTEX-COVER vu en classe. On peut développer un algorithme probabiliste qui retourne $X \subseteq V$ tel que $\mathbb{E}[|X|] =$

OPT_{LP}. L'idée est la suivante : pour chaque i de 1 à n, ajouter v_i à X avec probabilité x_i^*.

Argumentez d'abord que $\mathbb{E}[|X|] = OPT_{LP}$.

Ceci porte à croire que l'algorithme est une 1-approximation probabiliste. Il y a toutefois un problème. Quel est-il?

Solution.

Soit X notre solution. Soit I_v une variable indicatrice telle que $I_v = 1$ si $v \in X$ et $I_v = 0$ sinon. On a que

$$\mathbb{E}[APP] = \mathbb{E}[\sum_{v \in V} I_v] = \sum_{v \in V} \mathbb{E}[I_v] = \sum_{v \in V} x_v^* = OPT_{LP}$$

tel que demandé.

Le problème est qu'il n'y a aucune garantie que X soit une solution faisable. C'est-à-dire, il est possible que, par malchance, le X retourné ne couvre pas chaque arête, car ceci n'est pas vérifié par notre algorithme. Par exemple, il y a une probabilité non-nulle que l'on retourne $X = \emptyset$, qui n'est pas une solution faisable.

Exercice 7. Dans le problème SET-COVER-k-OCC, on a un univers U à couvrir avec un nombre minimum d'ensembles parmi $S = \{S_1, \ldots, S_m\}$. De plus, pour chaque $u \in U$, on peut supposer qu'il y a au plus k ensembles de S qui contiennent U.

Formulez ce problème avec un ILP. Donnez ensuite une k-approximation en relaxant ce ILP.

Indice. Si vous commencez avec $k = 2$, vous pourrez constater que c'est comme VERTEX-COVER. Généralisez ensuite à d'autres k.

Solution.

Pour chaque $S_i \in S$, on aura une variable x_i qui représente "S_i est dans notre solution".

Minimiser

$$\sum_{S_i \in S} x_i$$

Sujet à

$$\sum_{S_i : u \in S_i} x_i \geq 1 \qquad \text{pour tout } u \in U$$

$$x_i \in \{0, 1\} \qquad \text{pour tout } S_i \in S$$

La contrainte principale s'assure que pour chaque élément $u \in U$, on ait choisi au moins un des ensembles qui le contient. On obtient une relaxation LP en remplaçant la dernière contrainte par $0 \leq x_i \leq 1$.

Voici une k-approximation (rappelons que k est le nombre d'ensembles qui contiennent u):
– Résoudre le LP relaxation et obtenir une solution $\boldsymbol{x}^*$;
– Retourner $\{v \in V : x_v^* \geq 1/k\}$.

Bien sûr, $OPT \geq OPT_{LP}$. Pour APP, soit X la solution retournée par cet algorithme (donc X contient des ensembles). Prenons $u \in U$ et soient $S_{i_1}, \ldots, S_{i_k}$ les ensembles qui contiennent u. Notons que la contrainte $\sum_{S_i : u \in S_i} x_i \geq 1$ peut en fait s'écrire $x_{i_1} + \ldots + x_{i_k} \geq 1$. Il faut montrer que X est faisable, et donc que chaque $u \in U$ est couvert. À cause de cette contrainte, il faut qu'au moins un de $x_{i_1}^*, \ldots, x_{i_k}^*$ soit au moins $1/k$, car sinon $x_{i_1} + \ldots + x_{i_k}$ serait plus petit que 1. Donc, notre algorithme aura inclut un de $S_{i_1}, \ldots, S_{i_k}$ dans X. On déduit que u est couvert, et donc que X est une solution faisable.

Discutons maintenant d'approximation. Pour chaque $S_i \in X$, on a $x_i^* \geq 1/k$. On a donc

$$APP = |X| = \sum_{S_i \in X} 1 \leq \sum_{S_i \in X} k \cdot x_i^* \leq k \sum_{S_i \in S} x_i^* = k \cdot OPT_{LP} \leq k \cdot OPT$$

et c'est donc une k-approximation.

Exercice 8. Montrez que l'algorithme sacadosPoly des notes de cours (section 6.3) n'est pas toujours optimal. (sacadosPoly est l'algorithme qui divise toutes les valeurs par K et appelle la prog dynamique)

Solution.

Ce n'était pas si évident à trouver. L'idée est de montrer qu'on peut perdre l'optimal à cause des planchers. Voici l'instance la plus simple que j'ai pu trouver. Posons $\varepsilon = 1/2$ pour simplifier. On peut penser à une instance du style

$$(1, 2^n + 1), (1, 2^n + 1), (2, 2^{n+1}),$$

avec $W = 2$ (rappel : on a des paires (poids, valeur)). L'optimal est de choisir les deux objets $(1, 2^n + 1)$ pour une valeur de $2^{n+1} + 2$.

Par contre, pour arrondir, on divisera par $K = v_{max}/n \cdot \varepsilon = 2^{n+1}/(2n)$. L'instance modifiée devient donc

$$(1, \lfloor (2^n + 1)/2^{n+1} \cdot 2n \rfloor), (1, \lfloor (2^n + 1)/2^{n+1} \cdot 2n \rfloor), (2, \lfloor 2^{n+1}/2^{n+1} \cdot 2n \rfloor),$$

ce qui se simplifie à

$$(1, \lfloor n + 2n/2^{n+1} \rfloor), (1, \lfloor n + 2n/2^{n+1} \rfloor), (2, 2n)$$

Mais les planchers vont faire qu'on peut ignorer les termes $2n/2^{n+1}$. Donc l'instance est

$$(1, n), (1, n), (2, 2n)$$

Puisque les bris d'égalité sont arbitraires, il serait possible que le programmation dynamique retourne l'objet $(2, 2n)$ sur cette instance, ce qui est sous-optimal par rapport aux valeurs originales.

Avec un peu plus d'effort, on peut créer des instances sur lesquelles le programmation dynamique se trompe, peu importe comment le bris d'égalité est fait.
