Examen du cours d'approximation

{Emmanuelle.Lebhar, Nicolas.Schabanel}@ens-lyon.fr

Mercredi 7 janvier 2004 de 9h00 à 12h00

EXERCICE 1
On considère l'algorithme suivant pour Max-SAT : soit τ une instanciation arbitraire des variables et τ' son complémentaire (i.e., une variable est vraie dans τ' ssi elle est fausse dans τ) ; calculer les poids des clauses satisfaites par τ et τ', puis retourner la meilleure de ces deux instanciations.

Question 1 *Démontrez que cet algorithme est une $\frac{1}{2}$-approximation pour Max-SAT, puis exhibez une famille d'instance critique.*

EXERCICE 2
On considère le programme linéaire (primal) suivant :

$$\begin{aligned} \text{Maximiser} \quad & 2x_1 + 2x_2 - 2x_3 + x_4 \\ \text{sous les contraintes} \quad & 5x_1 + x_2 - 2x_3 = 3 \\ & -x_1 - x_2 + 4x_3 + x_4 \leq 2 \\ & x_1 + x_2 + 2x_3 - x_4 \geq 1 \\ & x_1 \geq 0,\ x_2 \leq 0,\ x_4 \geq 0 \end{aligned}$$

Question 2 *Donnez le dual de ce programme linéaire, et les conditions de complémentarité primal-dual associées. Et démontrez, de deux façons différentes, que les solutions $x = (\frac{4}{5}, 0, \frac{1}{2}, \frac{4}{5})$ et $y = (\frac{3}{5}, \frac{1}{5}, -\frac{4}{5})$ sont des solutions réalisables optimales du primal et du dual respectivement.*

On considère le programme linéaire (primal) suivant :

$$\begin{aligned} \text{Minimiser} \quad & 2x_1 + 3x_2 - x_3 \\ \text{sous les contraintes} \quad & 2x_1 - 3x_2 + x_3 \geq 1 \\ & -x_1 + 2x_2 - x_3 \geq 1 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{aligned}$$

Question 3 *Donnez les conditions de complémentarité primal-dual relâchées de paramètres $(\alpha, \beta) = (1, 2)$ associées (c'est-à-dire exact pour les conditions primales, mais à un facteur 2 pour les conditions duales). Et démontrez, de deux façons différentes, que les solutions $x = (7, 4, 0)$ et $y = (7, 12)$ sont des solutions réalisables du primal et du dual respectivement, à un facteur 2 de la valeur objectif optimale.*

EXERCICE 3

On étudie le problème de la couverture par sommets de poids minimum restreint aux graphes $G = (V, E)$, muni d'une fonction de poids w sur les sommets, et d'un k-coloriage valide des sommets (i.e., muni d'une fonction $c : V \to \{1, \ldots, k\}$ telle que pour tout $uv \in E$, $c(u) \neq c(v)$). La k-coloration est donnée mais n'est pas supposée optimale.

Question 4 *Proposez une $(2 - \frac{2}{k})$-approximation (polynomiale) pour ce problème. Et prouvez-le.*

Indication. Utilisez un programme linéaire pour la couverture par sommets de poids minimum dont toutes les solutions extrémales du problème relâché sont demi-entières (on ne redémontrera pas cette propriété) et remarquez que les sommets demi-entiers ne sont pas tous nécessaires dans la couverture.

Question 5 *Exhibez une famille d'instances critiques.*

EXERCICE 4

On étudie le problème suivant : vous décidez de vous mettre au ski, vous devez donc vous équipez, vous avez le choix entre louer le matériel ou bien l'acheter ; dans le premier cas, il vous en coûte x euros par jour, et dans le second, vous payez y euros une fois pour toute $(y > x)$; l'ennui de ce sport est qu'au bout d'un certain nombre de jours, τ, soit vous vous lasserez, soit vous serez irrémédiablement blessé ; vous ne connaissez pas cette date a priori, et de $t = 1$ jusqu'à cette date (incluse) vous faîtes du ski tous les jours. Le problème est de minimiser le coût du matériel de ski, i.e., de savoir quand vous vous décidez d'arrêter de louer pour acheter votre matériel (τ étant inconnu).

Ce problème est typiquement celui de la mise en veille du disque dur d'un ordinateur portable : le système cherche à minimiser la consommation d'énergie et ne sait pas quand l'utilisateur va réutiliser son disque, mais il doit décider quand le mettre en veille, ce qui lui coûtera par la suite un surcoût de consommation constant pour le remettre en route.

Question 6 *Supposez que $x = 3$ et $y = 10$, quelle est la stratégie optimale pour $\tau = 1$, $\tau = 3$, $\tau = 5$, $\tau = 10$?*

Question 7 *Posez $c = y/x$ (c est connu), et donnez la stratégie optimale en fonction de τ (qui vous est inconnu, mais que vous supposez connu par l'optimal).*

On se propose d'étudier la stratégie suivante, de paramètre t_0 : on loue tant que la date t est $\leq t_0$, puis on achète dès que $t > t_0$.

Question 8 *Donnez un facteur d'approximation pour cet algorithme (valable quelque soit la valeur τ, qui est inconnue).*

Question 9 *Démontrez que pour une valeur de t_0, cet algorithme est une 2-approximation (quelque soit la valeur τ, qui est inconnue). Quelle est cette valeur ? Peut-on obtenir un meilleur facteur d'approximation pour une autre valeur de t_0 ?*

EXERCICE 5

Étant donné n tâches $J_1, \ldots, J_n$ de durée d'exécution $p_1, \ldots, p_n$, un *ordonnancement* de ces tâches est une fonction qui à chaque tâche associe une date pour le début de son exécution. Deux tâches ne peuvent pas s'exécuter en même temps. On note $J_i \prec J_j$, si la tâche J_j dépend de la tâche J_i, i.e., si l'exécution de la tâche J_j ne peut commencer qu'après la fin de l'exécution de la tâche J_i.

On dit qu'un ordonnancement est *valide* si aucune tâche n'est ordonnancée en même temps qu'une autre et si chaque J_j est exécutée après la fin de l'exécution de J_i, pour tout $J_i \prec J_j$. On appelle *temps de complétion* C_i de la tâche J_i dans un ordonnancement donné, la date de la fin de l'exécution de la tâche dans cet ordonnancement.

On étudie le problème d'ordonnancement suivant :

Problème 1 (Ordonnancement avec dépendances) *Étant donné n tâches $J_1, \ldots, J_n$ de temps d'exécution $p_1, \ldots, p_n$, munies de poids $w_1, \ldots, w_n$ positifs, et un ensemble (acyclique) de relations de dépendance $J_i \prec J_j$, trouver un ordonnancement valide des tâches sur un processeur, qui minimise la somme pondérée des temps de complétion des tâches : $\sum_{i=1}^{n} w_i C_i$.*

Question 10 *Redémontrez rapidement la règle de Smith qui affirme qu'en l'absence de dépendance, un ordonnancement optimal est obtenu en ordonnançant les tâches par ratio w_i/p_i décroissant.*

Indication. On pourra étudier l'effet de l'inversion de deux tâches.

Nous étudions maintenant le cas général avec dépendance.

Question 11 *Démontrez que dans tout ordonnancement valide des tâches (avec ou sans dépendance),*

$$\sum_{i=1}^{n} p_i C_i \geq \sum_{i=1}^{n} p_i \sum_{j=1}^{i} p_j$$

Indication. Appliquez la règle de Smith sur l'instance modifiée suivante : aucune dépendance et pour fonction de poids $w'_i = p_i$; et remarquez que dans ce cas, l'ordre des tâches est sans importance.

Question 12 *Démontrez de même que pour tout ordonnancement valide des tâches,*

$$\forall A \subseteq \{1, \ldots, n\}, \quad \sum_{i \in A} p_i C_i \geq \frac{1}{2} \sum_{i \in A} p_i^2 + \frac{1}{2} \Big(\sum_{i \in A} p_i \Big)^2$$

Indication. Remarquez que : $\displaystyle\sum_{i=1}^{n} p_i \sum_{j=1}^{i} p_j = \frac{1}{2} \sum_{i=1}^{n} p_i^2 + \frac{1}{2} \Big(\sum_{i=1}^{n} p_i \Big)^2$.

On considère donc le programme linéaire suivant (qui admet un nombre exponentiel de contraintes, mais qui, malgré tout, se résout exactement en temps polynomial par la méthode des ellipsoïdes). On note y_i la variable associée au temps de complétion de la tâche J_i dans le programme linéaire.

$$\begin{aligned} \text{Minimiser} \quad & \sum_{i=1}^{n} w_i y_i \\ \text{sous les contraintes} \quad & \sum_{i\in A} p_i y_i \geq \frac{1}{2}\sum_{i\in A} p_i^2 + \frac{1}{2}\Big(\sum_{i\in A} p_i\Big)^2 \quad \forall A \subseteq \{1,\ldots,n\} \\ & y_i \geq 0 \qquad\qquad\qquad\qquad \forall i \end{aligned}$$

On peut démontrer que les solutions extrémales de ce problème définissent des ordonnancements valides sans dépendance entre les tâches ; cependant ce n'est plus le cas lorsqu'on introduit des dépendances entre les tâches.

Question 13 *Pour tenir compte des dépendances, définissez pour chaque dépendance $J_i \prec J_j$, une contrainte liant les variables y_i et y_j. Donnez le nouveau programme linéaire incluant les dépendances. Démontrez que ce programme linéaire est un minorant du temps de complétion pondéré optimal.*

Étudions l'algorithme suivant : soit y^* une solution optimale (calculée en temps polynomial) du programme linéaire complet (avec les contraintes de dépendance) ; y^* ne définit pas en général un ordonnancement valide, mais on ordonnance les tâches dans l'ordre défini par y^* (les tâches ayant le même y_i^* sont ordonnancées dans un ordre arbitraire) ; notons S l'ordonnancement obtenu.

Question 14 *Démontrez que les contraintes de dépendance sont satisfaites dans S.*

Quitte à renuméroter les tâches, supposons que $y_1^* \leq \ldots \leq y_n^*$. Notons $C_i = \sum_{j=1}^{i} p_j$, le temps de complétion de J_i dans S.

Question 15 *Démontrez que, pour tout i,*

$$y_i^* \geq \frac{1}{2} C_i.$$

Indication. Utilisez les contraintes du programme linéaire.

Question 16 *Démontrez que cet algorithme est une 2-approximation.*

Question 17 *Exhibez une famille d'instances critiques pour cet algorithme.*

Indication. Considérez n tâches de temps d'exécution unitaire $J_1, \ldots, J_n$ telles que pour tout $i < n$, $J_i \prec J_n$. On munit ces tâches des poids $w_1 = \cdots = w_{n-1} = 1$ et $w_n = M$. Pour M assez grand, on a $y_1^* = \cdots = y_{n-1}^* = \frac{n+1}{2} - \frac{1}{n}$ et $y_n^* = \frac{n+3}{2} - \frac{1}{n}$.

On suppose maintenant que chaque tâche n'est disponible qu'à partir d'une date r_i donnée en entrée, i.e., J_i ne peut pas être ordonnancée avant la date r_i.

Question 18 *Quelles contraintes doit-on ajouter au programme linéaire pour tenir compte de la disponibilité des tâches ?*

Question 19 *Proposez un algorithme qui construit un ordonnancement pour ce nouveau problème.*

Question 20 (★) *Démontrez que cet algorithme est une* 3*-approximation.*

EXERCICE 6

L'objectif de cet exercice est de démontrer que, pour tout $\epsilon > 0$, il n'existe pas de $(\frac{7}{8}+\epsilon)$-approximation pour Max-3SAT, à moins que $\mathbf{P} = \mathbf{NP}$.

Nous définissons la classe $\mathbf{PCP}_{c,s}(r(n), q(n))$ comme suit : un langage L appartient à $\mathbf{PCP}_{c,s}(r(n), q(n))$ s'il existe un vérifieur (polynomial) V pour L, utilisant $O(r(n))$ bits et lisant $O(q(n))$ bits de la preuve y, tel que :

- si $x \in L$, il existe une preuve y tel que V accepte (x, y), avec probabilité $\geq c$,
- si $x \notin L$, alors pour toute preuve y, V accepte (x, y) avec probabilité $< s$,

les probabilités étant prises sur la chaîne de bits aléatoires r de longueur $O(r(n))$.

Nous admettons le résultat suivant (dû à Haståd, 1997) :

Théorème 1 *Pour tout* $\epsilon > 0$*,*

$$\mathbf{NP} = \mathbf{PCP}_{1-\epsilon,\frac{1}{2}+\epsilon}(\log n, 1).$$

De plus, il existe un vérifieur particulièrement simple pour SAT dans cette classe, qui utilise $c \log n$ *bits aléatoires pour calculer trois positions seulement de la preuve* y*, disons* i*,* j*, et* k*, et un bit* b*, et qui accepte* (x, y) *ssi :*

$$y_i + y_j + y_k \equiv b \pmod 2,$$

où y_i *désigne le* i*-ème bit de la preuve* y*. Nous noterons* V_{SAT} *ce vérifieur.*

À l'instar de la réduction vu en cours, nous allons utiliser un problème intermédiaire pour construire notre réduction séparatrice de SAT à Max-3SAT. Ce problème est le suivant, noté Max-3EQ :

Problème 2 (Systèmes d'équations linéaires à 3 variables dans $\mathbb{Z}/2\mathbb{Z}$**)**
Étant données m *équations linéaires, utilisant chacune trois variables parmi* $x_1, \ldots, x_n$ *à valeurs dans* $\mathbb{Z}/2\mathbb{Z}$*, trouvez une instanciation des variables qui maximise le nombre d'équations linéaires satisfaites.*

Rappel. Une équation à trois variables x, y, et z dans $\mathbb{Z}/2\mathbb{Z}$ est nécessairement de la forme $x + y + z = b$, où $b = 0$ ou 1.

Question 21 *Exhibez à partir du vérifieur* V_{SAT} *donné par le théorème 1, une réduction séparatrice (polynomiale) de* SAT *à* Max-3EQ $\Psi : \varphi \mapsto \Sigma$ *telle que, si* m' *désigne le nombre d'équations dans le système* $\Sigma = \Psi(\varphi)$ *:*

- *si* φ *est satisfiable, alors il existe une instanciation qui satisfait l'ensemble des* m' *équations de* Σ*, i.e.,* Max-3EQ$(\Sigma) = m'$*, et*
- *si* φ *n'est pas satisfiable, alors* Max-3EQ$(\Sigma) < (\frac{1}{2} + \epsilon)m'$*.*

Indication. Inspirez-vous de la réduction séparatrice vu en cours pour Max-SAT : indexez les équations utilisées par le vérifieur par toutes les chaînes aléatoires possibles, pour construire l'instance réduite.

Question 22 *Déduisez que, pour tout $\epsilon > 0$, il n'existe pas de $(\frac{1}{2}+\epsilon)$-approximation pour* Max-3EQ*, à moins que* $\mathbf{P} = \mathbf{NP}$.

Question 23 *Exhibez une gap-preserving réduction (polynomiale) de* Max-3EQ *à* Max-3SAT *qui démontre que, pour tout $\epsilon > 0$, il n'existe pas de $(\frac{7}{8}+\epsilon)$-approximation pour* Max-3SAT*, à moins que* $\mathbf{P} = \mathbf{NP}$.

Indication. Chaque équation $x + y + z = 0 \pmod 2$ peut se réécrire sous la forme de quatre 3-clauses :

$$(\overline{x} \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee \overline{z}),$$

où $\overline{x}$ désigne la négation de x.