Exercices d'algorithmes d'approximation.

Chapitres 1 et 2.

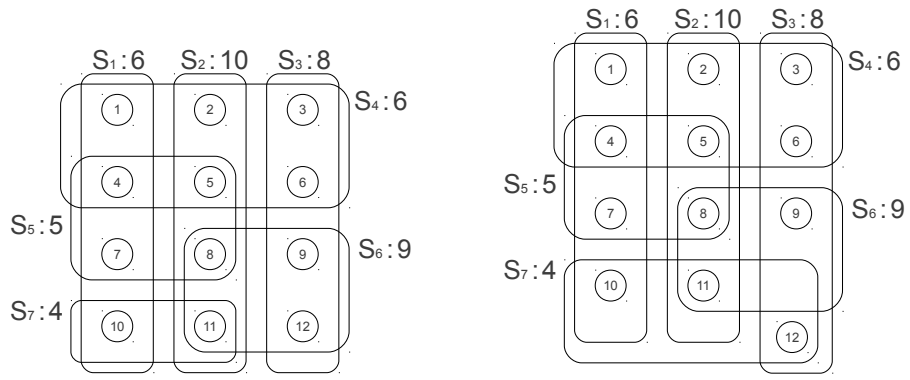
Absil Romain

6 avril 2011

Exercice 1. Soit $\mathcal{P}_1$ et $\mathcal{P}_2$ deux instances du problème SET-COVER, illustrées à la Figure 1.

1. Écrivez $\mathcal{P}_2$ sous la forme primale d'un problème d'optimisation linéaire en nombres entiers.
2. Écrivez la forme duale de la relaxation linéaire de ce problème.
3. Calculez la valeur du paramètre f utilisé dans les algorithmes d'approximation DETERMINIST-ROUNDING-SC, DUAL-ROUNDING-SC et PRIMAL-DUAL-SC.
4. Utilisez l'arrondi déterministe DETERMINIST-ROUNDING-SC sur la solution de la relaxation linéaire du problème primal et vérifiez son facteur d'approximation. Calculez également le paramètre α afin de vérifier la garantie *a fortiori* de cet algorithme.
5. Utilisez la méthode d'arrondi déterministe DUAL-ROUNDING-SC sur la solution du problème dual pour résoudre ce problème et vérifiez son facteur d'approximation.
6. Utilisez l'algorithme primal-dual PRIMAL-DUAL-SC pour résoudre ce problème et vérifiez son facteur d'approximation.
7. Utilisez l'algorithme GREEDY-SC pour résoudre $\mathcal{P}_1$.
8. Calculez le paramètre g pour cette méthode et vérifiez son facteur d'approximation.

Remarque : vous ne devez *pas* calculer vous-même les solutions des formulations primales et duales du problème. Elles vous seront données en séance d'exercices.



(a) $\mathcal{P}_1$ (b) $\mathcal{P}_2$

FIGURE 1 – Deux instances $\mathcal{P}_1$ et $\mathcal{P}_2$ du problème SET-COVER

Solution.

1. Formulation primale :

$$\min \; 6x_1 + 10x_2 + 8x_3 + 6x_4 + 5x_5 + 9x_6 + 4x_7$$

$$\begin{array}{lllll}
\text{s.c.} & v_1 & : & x_1 + x_4 & \geqslant 1 \\
& v_2 & : & x_2 + x_4 & \geqslant 1 \\
& v_3 & : & x_3 + x_4 & \geqslant 1 \\
& v_4 & : & x_1 + x_4 + x_5 & \geqslant 1 \\
& v_5 & : & x_2 + x_4 + x_5 & \geqslant 1 \\
(& v_6 & : & x_3 + x_4 & \geqslant 1 \quad) \\
& v_7 & : & x_1 + x_5 & \geqslant 1 \\
& v_8 & : & x_2 + x_5 + x_6 & \geqslant 1 \\
& v_9 & : & x_3 + x_6 & \geqslant 1 \\
& v_{10} & : & x_1 + x_7 & \geqslant 1 \\
& v_{11} & : & x_2 + x_6 + x_7 & \geqslant 1 \\
& v_{12} & : & x_3 + x_7 & \geqslant 1 \\
& & & x_i \in \{0,1\} \quad \forall i = 1, \ldots, 7 &
\end{array}$$

2. Formulation duale :

$$\max \sum_{i=1}^{12} v_i$$

$$\begin{array}{lllll}
\text{s.c.} & s_1 & : & v_1 + v_4 + v_7 + v_{10} & \leqslant 6 \\
& s_2 & : & v_2 + v_5 + v_8 + v_{11} & \leqslant 10 \\
& s_3 & : & v_3 + v_6 + v_9 + v_{12} & \leqslant 8 \\
& s_4 & : & v_1 + v_2 + v_3 + v_4 + v_5 + v_6 & \leqslant 6 \\
& s_5 & : & v_4 + v_5 + v_7 + v_8 & \leqslant 5 \\
& s_6 & : & v_8 + v_9 + v_{11} & \leqslant 9 \\
& s_7 & : & v_{10} + v_{11} + v_{12} & \leqslant 4 \\
& & & v_i \geqslant 0 \quad \forall i = 1, \ldots, 12 &
\end{array}$$

3. Avant toute chose, on remarque que la solution optimale du problème a une valeur de 23 en sélectionnant les ensembles S_3, S_4, S_5 et S_7.

   Par ailleurs, pour les trois premiers algorithmes, on a

$$\begin{aligned} f &= \max_i f_i \\ &= \max_i \Big| \{ j : v_i \in S_j \} \Big| \end{aligned}$$

   Plus particulièrement, on a $f = 3$ (atteint sur les éléments v_4, v_5, v_8 et v_{11}).

   On pourra vérifier les résultats théoriques de ce facteur d'approximation sur les trois algorithmes.

4. Dans la méthode d'arrondi déterministe, on doit utiliser la règle d'arrondi suivante sur une solution x^* :

$$x_i = \begin{cases} 1 & \text{si } x_i^* \geqslant \frac{1}{f}, \\ 0 & \text{sinon.} \end{cases}$$

   Comme la solution du problème primal relaxé vaut 22, avec

$$x^* = \left(\frac{1}{2}, 0, \frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right), \text{ [1]}$$

   à l'aide de cette règle, on obtient donc la solution $(1, 0, 1, 1, 1, 1, 1)$.

1. Vous ne devez pas chercher vous-même cette solution, elle a été donnée en séance d'exercices.

La garantie α *a fortiori* de cet algorithme est définie comme

$$\alpha = \frac{\sum_{i \in I} w_i}{z^*_{LP}}.$$

On a donc $\alpha = \frac{6+8+6+5+9+4}{22} = \frac{38}{22}$.

On remarque par ailleurs qu'on a $\frac{APP}{OPT} = 1.6521 < 1.7272 = \alpha < 3 = f$.

5. La règle d'arrondi dual est la suivante : *On sélectionne j dans I' si l'inégalité correspondante est serrée,* i.e., *si*

$$\sum_{i : e_i \in S_j} y^*_i = w_j.$$

Comme la solution du problème dual vaut 22, avec

$$v^* = (0, 6, 0, 0, 0, 0, 3, 2, 7, 3, 0, 1), \text{ [2]}$$

on sélectionne donc tous les ensembles à l'exception de S_2. Cette solution a une valeur d'objectif de $38 < f.OPT = 69$.

6. Pour l'algorithme PRIMAL-DUAL-SC, on notera NC l'ensemble des sommets non couverts à l'itération courante. À chaque itération, on calcule les variables nécessaires à l'algorithme. Les notations utilisées sont identiques à celles du cours.
   – Itération 1.

$$NC = \left\{ \boxed{v_1}, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12} \right\}$$

$s_1 : \Delta_1 = 6$
$s_4 : \Delta_4 = 6$
$\hookrightarrow l = 1 \quad$ (ou 4)

$y = (6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$
$I = \{1\}$

2. Vous ne devez pas chercher vous-même cette solution, elle a été donnée en séance d'exercices.

– Itération 2.

$$NC = \left\{ \boxed{v_2}, v_3, v_5, v_6, v_8, v_9, v_{11}, v_{12} \right\}$$

$s_2 : \Delta_1 = 10$
$s_4 : \Delta_4 = 0$
$\hookrightarrow l = 4$

$y = (6, \underline{0}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$
$I = \{1, 4\}$

– Itération 3.

$$NC = \left\{ \boxed{v_8}, v_9, v_{11}, v_{12} \right\}$$

$s_2 : \Delta_2 = 10$
$s_5 : \Delta_5 = 5$
$s_6 : \Delta_6 = 9$
$\hookrightarrow l = 5$

$y = (6, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0)$
$I = \{1, 4, 5\}$

– Itération 4.

$$NC = \left\{ \boxed{v_9}, v_{11}, v_{12} \right\}$$

$s_3 : \Delta_3 = 8$
$s_6 : \Delta_6 = 4$
$\hookrightarrow l = 6$

$y = (6, 0, 0, 0, 0, 0, 0, 5, 4, 0, 0, 0)$
$I = \{1, 4, 5, 6\}$

– Itération 5.

$$NC = \left\{ \boxed{v_{12}} \right\}$$

$s_3 : \Delta_3 = 4$
$s_7 : \Delta_7 = 4$
$\hookrightarrow l = 3$

$y = (6, 0, 0, 0, 0, 0, 0, 5, 4, 0, 0, 4)$
$I = \{1, 3, 4, 5, 6\}$

Les ensembles sélectionnées sont donc S_1, S_3, S_4, S_5 et S_6, avec un objectif de valeur $34 < f.OPT = 69$.

7. À chaque itération, l'algorithme GREEDY-SC sélectionne l'ensemble ayant le meilleur rapport éléments non-couverts – coût de l'ensemble. Pour cet exercice, on notera Q_i ce rapport pour chaque ensemble S_i. On a donc les résultats suivants :

Itération	1	2	3	4
Q_1	$\frac{6}{4}$	$\frac{6}{2}$	$\frac{6}{1}$	/
Q_2	$\frac{10}{4}$	$\frac{10}{2}$	$\frac{10}{1}$	/
Q_3	$\frac{8}{4}$	$\frac{8}{2}$	$\frac{8}{2}$	$\boxed{\frac{8}{2}}$
Q_4	$\boxed{\frac{6}{6}}$	/	/	/
Q_5	$\frac{5}{4}$	$\frac{5}{2}$	$\boxed{\frac{5}{2}}$	/
Q_6	$\frac{9}{4}$	$\frac{9}{4}$	$\frac{9}{3}$	$\frac{9}{2}$
Q_7	$\frac{4}{2}$	$\boxed{\frac{4}{2}}$	/	/
Sélection	S_4	S_7	S_5	S_3

L'algorithme sélectionne donc les ensembles S_3, S_4, S_5 et S_7, avec un objectif de valeur 23. La solution optimale a une valeur de 21 et est atteinte en sélectionnant S_1, S_4 et S_6.

8. L'algorithme GREEDY-SC est un algorithme de H_g-approximation, où $g = \max_i |S_i|$.

   Dans le cas de $\mathcal{P}_1$, on a $g = 6$, et donc GREEDY-SC est un algorithme de 2.45-approximation.

□

Exercice 2. Montrez que l'Algorithme 1 est un algorithme de 2-approximation pour le problème VERTEX-COVER.

Solution.
Cette heuristique est clairement polynomiale. En notant C^* une couverture optimale, montrons que $|C| \leqslant 2\ |C^*|$.

L'ensemble C des sommets sélectionnés est bel et bien une couverture, car l'algorithme s'arrête quand toute arête de G a été couverte par un sommet dans C.

Algorithm 1 Algorithme APPROX-VERTEX-COVER

Entrées : $G = (V, E)$ un graphe non orienté.
Sortie : C : une couverture des arêtes par les sommets de G.

1: $C \leftarrow \emptyset$
2: $E' \leftarrow E$
3: **tant que** $E' \neq \emptyset$
4: Soit (u, v) une arête arbitraire de E'
5: $C \leftarrow C \cup \{u, v\}$
6: Supprimer de E' toutes les arêtes incidentes à u ou v
7: **retourner** C

Afin de montrer que APPROX-VERTEX-COVER calcule une couverture de taille au plus deux fois la taille d'une couverture optimale, notons A l'ensemble des arêtes sélectionnées à la ligne 4 de cet algorithme. On remarque que toute couverture – et en particulier une couverture optimale C^* – doit inclure une des extrémités de chaque arête de A. De plus, il n'existe pas d'arêtes de A ayant une extrémité commune, car lorsqu'une arête est sélectionnée à la ligne 4, toutes les arêtes incidentes sont supprimées de E' à la ligne 6.

Dès lors, deux arêtes de A ne sont jamais couvertes par le même sommet de C^*, et on a donc $|C^*| \geqslant |A|$. Par ailleurs, chaque exécution de la ligne 4 sélectionne une arête dont aucune des extrémités ne se trouve dans C, et on obtient donc $|C| = 2\,|A|$. En combinant ces deux équations, on obtient donc

$$\begin{aligned} |C| &= 2\,|A| \\ &\leqslant 2\,|C^*| \end{aligned}$$

□

Exercice 3. Un voleur veut cambrioler une maison, muni d'un sac à dos de taille K. Dans la maison se trouve une série de n objets qu'il souhaite dérober. Chaque objet e_i a une valeur v_i et une taille s_i.
Le but du voleur est donc de remplir son sac à dos avec le plus d'objets possibles afin de maximiser son profit, en sachant qu'il ne peut pas remplir son sac au delà de sa capacité.
En optimisation combinatoire, on appelle ce problème le *problème de sac à dos binaire* (0-1-Knapsack Problem). Si le voleur a le droit de sélectionner autant de fois qu'il le désire un même objet, on appellera ce problème le *problème de sac à dos entier* (Integer-Knapsack Problem).

1. Modélisez le problème de sac à dos binaire sous la forme d'un problème d'optimisation en nombres entiers,
2. Décrivez une heuristique greedy pour ce problème,
3. Montrez que cette heuristique est un algorithme de 2-approximation (ou modifiez-là pour qu'elle le soit),

4. Décrivez une heuristique greedy pour le problème de sac à dos entier et montrez que c'est un algorithme de 2-approximation.

Solution.

1. On peut modéliser le problème de sac à dos binaire de la façon suivante :

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} v_i \; x_i \\ s.c. \quad & \sum_{i=1}^{n} s_i \; x_i < K \\ & x_i \in \{0,1\} \; \forall i = 1, \ldots, n \end{aligned}$$

   Les variables x_i décrivent si l'on a sélectionné l'objet e_i ou non.

2. Une heuristique Greedy va itérativement sélectionner les objets de profit maximal. Le profit d'un objet i peut être décrit par $\frac{v_i}{s_i}$. On peut donc écrire l'heuristique suivante pour ce problème :

Algorithm 2 Algorithme Greedy

Entrées : un problème de sac à dos binaire.
Sortie : z^*_{GR}, solution greedy pour le problème.

1: Trier et renuméroter les objets i par ordre de profit $\frac{v_i}{s_i}$ décroissant.
2: Ajouter dans l'ordre les objets i dans le sac tant que la capacité du sac n'est pas dépassée.
3: **retourner** le contenu du sac à dos.

3. L'heuristique `Greedy` n'est pas un algorithme d'approximation, et la solution peut être aussi mauvaise que possible, comme illustré par l'exemple suivant :
   - Un objet de taille 1 et de valeur 2,
   - Un objet de taille K et de valeur K.

   L'algorithme `Greedy` va uniquement sélectionner le premier objet, et en faire une approximation de mauvaise qualité. Afin d'éviter ce problème et de transformer cette heuristique en algorithme d'approximation, il est nécessaire d'y apporter la modification suivante : soit k le dernier objet ajouté dans le sac, il suffit de retourner $\max \left\{ \sum_{i=1}^{k} v_i, v_{k+1} \right\}$.

Montrons qu'à présent cette heuristique est bien un algorithme de 2-approximation, *i.e.,* l'algorithme est polynomial, et la solution retournée vaut au pire la moitié de la solution optimale.

Clairement, cet algorithme a une complexité dans le pire des cas en $\mathcal{O}(n \log n)$. En notant z^* la solution optimale et z^*_{GR} la solution de l'algorithme, on doit donc montrer que $z^* \leqslant 2\ z^*_{GR}$.

Avec cette heuristique, si la solution n'est pas optimale il reste de la place dans le sac. Précisément, il reste un espace de taille $K - \sum_{i=1}^{k} v_i$. Dès lors, on pourrait améliorer la solution en ajoutant des morceaux de l'objet $k+1$ (cette solution n'est évidemment pas admissible).

On a donc

$$\begin{aligned} z^* &\leqslant \sum_{i=1}^{k} v_i + \left(K - \sum_{i=1}^{k} v_i\right) \cdot \frac{v_{k+1}}{s_{k+1}} \\ &< \sum_{i=1}^{k} v_i + v_{k+1} \end{aligned}$$

Si $\sum_{i=1}^{k} v_i \geqslant v_{k+1}$, alors $z^* \leqslant 2 \sum_{i=1}^{k} v_i = 2\ z^*_{GR}$.

Sinon si $\sum_{i=1}^{k} v_i < v_{k+1}$, alors $z^* \leqslant 2\ p_{k+1} = 2\ z^*_{GR}$.

4. L'heuristique greedy pour ce problème procède de manière similaire à l'heuristique du point 2 : elle trie dans un premier temps les objets par profit $\frac{v_i}{s_i}$ décroissants, et sélectionne ensuite le premier objet $\left\lfloor \frac{K}{s_1} \right\rfloor$ fois.

   Montrons à présent que c'est un algorithme de 2-approximation, *i.e.,* que cette heuristique est polynomiale (trivial) et que la solution retournée vaut au pire la moitié de la solution optimale.

   On a $z^*_{GR} = v_1 \left\lfloor \frac{K}{s_1} \right\rfloor$. Notons z_{LP} la solution de la relaxation linéaire. On a :

$$\begin{aligned} z^* &\leqslant z_{LP} && \text{par définition de relaxation} \\ &= v_1 \tfrac{K}{s_1} \\ &= v_1 \left(\left\lfloor \tfrac{K}{s_1} \right\rfloor + f \right) && \text{pour un certain } 0 \leqslant f < 1 \\ &\leqslant 2 v_1 \left\lfloor \tfrac{K}{s_1} \right\rfloor \end{aligned}$$

On a donc $\frac{z^*_{GR}}{z^*} \geqslant \frac{z^*_{GR}}{z_{LP}} \geqslant \frac{1}{2}$.

□