MIM2 - Algorithmes d'approximation- TD 7
Emmanuelle Lebhar
elebhar@ens-lyon.fr - Bureau 317

FPTAS et sac-à-dos
3 Novembre 2003

Exercice 1 (Sur l'agorithme glouton du sac-à-dos)

On étudie l'algorithme glouton qui trie les objets par ratio valeur/taille croissant et les sélectionne dans cet ordre de manière gloutonne.

1. Nous avons vu en cours que cet algorithme peut donner des résultats arbitrairement mauvais. Quel problème localisez-vous ?
2. On note $\mathrm{val}_{gl}(x)$ la valeur de la solution retournée par le glouton sur l'instance x et $p_{\max}$ la plus grande valeur parmi les objets. Soit $\mathrm{val}_{\mathrm{new}}(x) = \max(p_{\max}, \mathrm{val}_{gl}(x))$. Montrez que $\mathrm{val}(OPT)/\,\mathrm{val}_{\mathrm{new}}(x) < 2$.
3. En déduire une amélioration de performance pour l'algorithme.

Exercice 2 (NP-difficulté au sens fort)

Un problème Π est dit *NP-difficile au sens fort* si pour tout problème Π' de NP, il existe une réduction polynomiale de Π' à Π telle que les nombres soient tous écrits en unaire dans le problème réduit.

Démontrez qu'aucun problème NP-difficile au sens fort n'admet d'algorithme pseudo-polynomial, à moins que P=NP.

Exercice 3 (FPTAS pour partition)

On considère le problème suivant.

Ensembles somme-ratio minimaux : (*subset-sum ratio problem* en anglais) étant donnés n entiers positifs $0 < a_1 < \ldots < a_n$, déterminer deux sous-ensembles disjoints non vides $S_1, S_2 \subseteq \{1, \ldots n\}$ tels que $\sum_{i\in S_1} a_i \geq \sum_{i\in S_2} a_i$, et tels que le ratio $\sum_{i\in S_1} a_i / \sum_{i\in S_2} a_i$ soit minimal.

On note $B = \sum_{i=1}^{n} a_i$.

– *Algorithme pseudo-polynomial :*

1. Nous allons commencer par donner un algorithme pseudo-polynomial pour ce problème. Pour cela, on introduit deux tableaux : $t[0 \ldots n, 0 \ldots B]$ dont les entrées sont dans $\{0, 1\}$ et $c[0 \ldots n, 0 \ldots B]$ dont les entrées sont des sous-ensembles de $\{1, \ldots n\}$.
   – $t[i, j] = 1$ ssi il existe un ensemble $S \subseteq \{1, \ldots, i\}$ avec $i \in S$ et $\sum_{k\in S} a_k = j$.

- $c[i, j] = S$ où S est le sous-ensemble satisfaisant les conditions ci-dessus s'il existe ; $S = \emptyset$ sinon.

   Ecrivez un algorithme qui remplit ces tableaux.
2. Montrez que s'il existe deux indices $i_1 \neq i_2$ tels que $t[i_1, j] = t[i_2, j] = 1$, alors on peut s'arrêter.
3. Dans le cas contraire, montrez que la solution optimale est $\inf\{l/j : \exists(i, k), t[i, j] = t[k, l] = 1 \text{ et } c[i, j] \cap c[k, l] = \varnothing\}$.
4. Montrez que cet algorithme est bien pseudo-polynomial.

- *FPTAS :*
  1. Le schéma d'approximation va s'éxécuter sur toutes les instances I_m pour $2 \leq m \leq n$ réduites aux éléments $a_1 < \ldots < a_m$ et prendre ensuite la meilleure des solutions. Considérons une de ces instances I_m et posons, pour $0 < \varepsilon < 1$, $k(m) = \varepsilon^2 \cdot a_m/(2m)$. Soit $n_0 \leq n$ le plus grand indice tel que $k(n_0) < 1$. Que peut-on faire pour $m \leq n_0$?(**cas 1**)
  2. On considère maintenant $m > n_0$ (**cas 2**). Nous allons transformer I_m en une instance I'_m ne contenant que des éléments de taille polynomiale. Que proposez-vous pour définir les a'_i ? L'instance I'_m est constituée des éléments a'_i tels que $a'_i \geq m/\varepsilon$?
  3. Supposons qu'une instance I'_m contienne t éléments. Remarquez que I'_m est non-vide. Que peut-on faire lorsque $t = 1$? (**cas 2.1**)
  4. Supposons $t > 1$ (**cas 2.2**). On applique l'algroithme pseudo-polynomial. Que peut-on dire lorsque $\mathrm{OPT}(I'_m) = 1$ (**cas 2.2.1**) ?

     Plaçons-nous dans le cas où $\mathrm{OPT}(I'_m) \neq 1$ (**cas 2.2.2**). Voici le principe dans ce cas. On construit 3^{t-1} paires d'ensembles disjoints $P(\tilde{v}, m), Q(\tilde{v}, m)$, inclus dans l'ensemble des indices de I'_m. Ils sont paramétrés par un vecteur $\tilde{v} \in \{0, 1, 2\}^{t-1}$ qui caractérise les indices présents dans les deux ensembles.
     Si $\sum_{i \in P} a_i > \sum_{i \in Q} a_i$, on pose $R_1 = P$ et $R_2 = Q$, et inversement. Soit j le plus petit indice tel que :
     $$\sum_{i \in R_2} a_i + \sum_{i=j+1}^{m-t} a_i < \sum_{i \in R_1} a_i$$
     On défini alors les deux ensembles S_1 et S_2 : si $j = 0$, $S_1 = R_1$ et $S_2 = R_2 \cup \{1, \ldots, m-t\}$ Sinon, pour $m \in R_2$, même affectation et pour $m \in R_1$, $S_1 = R_2 \cup \{1, \ldots, m-t\}$ et $S_2 = R_1$.
     L'algorithme prend ensuite le vecteur $\tilde{v}$ qui minimise le ratio $\sum_{i \in S_1} a_i / \sum_{i \in S_2} a_i$ et retourne S_1, S_2 comme solution.
  5. Montrez que l'algorithme présenté est une $(1 + \varepsilon)$-approximation en temps polynomial en n et en ε.