Corrigé de l'Examen d'optimisation combinatoire, M1 MIAGE
Classique
Durée 2h - le 11 avril 2008 -Aucun document autorisé

Exercice 1 Question de cours

1 Qu'est-ce qu'une relaxation d'un problème d'optimisation, et quelle propriété en a la solution optimale, selon la nature de l'optimisation (minimisation ou maximisation) ?

Si l'on considère un problème $\max_{x \in D} f(x)$, la relaxation consiste à définir un domaine D' tel que $D \subset D'$, et à calculer, si cela est plus simple, $\max_{x \in D'} f(x) \geq \max_{x \in D} f(x)$. On obtient donc un majorant de la solution optimale recherchée. Dans le cas d'une minimisation, on a $\min_{x \in D'} f(x) \leq \min_{x \in D} f(x)$

2 Qu'est-ce qu'un algorithme avec une approximation relative α pour un problème de minimisation ?

En considérant un problème $\min_{x \in D} f(x)$, et un algorithme $\mathcal{A}$ qui fournit une solution $x_{\mathcal{A}} \in D$, si l'on note x_{opt} la solution optimale du problème, l'algorithme $\mathcal{A}$ a une approximation relative de α si pour toute instance du problème,

$$f(x_{\mathcal{A}}) \leq \alpha f(x_{opt})$$

Exercice 2 Binômes

On considère un ensemble de personnes $P = \{1, \ldots, n\}$ qui doivent travailler en binôme sur des projets (on supposera que n est pair). Chaque personne possède une capacité c_i connue à l'avance et qui mesure son efficacité au travail : plus c_i est petit, plus la personne est rapide dans l'exécution de sa part du projet. Si on constitue un binôme entre deux personnes i et j alors le temps que mettra le binôme pour terminer son projet est $c_i + c_j$.

On dispose de plus d'un graphe d'incompatibilité $G = (P, A)$, dont les sommets sont les personnes. L'existence d'une arête $\{i, j\}$ signifie que les personnes i et j ne peuvent pas être en binôme.

Le but est de répartir les personnes en binômes compatibles de sorte que tous les projets soient terminés en un minimum de temps (Les binômes travaillent en parallèle).

1 Modéliser ce problème à l'aide d'un programme mathématique.

Deux solutions parmi les solutions possibles :

Première solution :

Inconnues : $x_{ij}, i, j \in P$. x_{ij} vaut 1 si les personnes i et j forment un binôme et 0 sinon.

Contraintes :

a)les personnes incompatibles ne forment pas de binôme : $\forall \{i,j\} \in A, \quad x_{ij} = 0$

b) les binômes sont symétriques : $\forall i, j \in P, \quad x_{ij} = x_{ji}$

c) Chaque personne fait partie d'un seul binôme : $\forall i \in P, \sum_{j=1}^{n} x_{ij} = 1$

d)$\forall i, j \in P, \quad x_{ij} \in \{0,1\}$

Objectif : la durée de travail d'un binôme entre i et j est $(c_i + c_j)x_{ij}$. Par conséquent, la date à laquelle tous les projets sont terminés est $\max_{i,j \in P}(c_i + c_j)x_{ij}$. Il faut donc minimiser cette quantité :

$$\min(\max_{i,j \in P}(c_i + c_j)x_{ij})$$

Seconde solution :

Inconnues : $x_{ik}, i \in P, k \in \{1, \ldots, \frac{n}{2}\}$. x_{ik} vaut 1 si la personne i fait partie du binôme k, 0 sinon.

Contraintes :

a)les personnes incompatibles ne sont pas dans un même binôme : $\forall k \in \{1, \ldots, \frac{n}{2}\} \forall \{i,j\} \in A, \quad x_{ik} + x_{jk} \leq 1$

b) il y a deux personnes dans un binôme $\forall k \in \{1, \ldots, \frac{n}{2}\} \quad \sum_{i=1}^{n} x_{ik} = 2$

c) Chaque personne fait partie d'un seul binôme : $\forall i \in P, \sum_{k=1}^{n} x_{ik} = 1$

d)$\forall k \in \{1, \ldots, \frac{n}{2}\} \forall i \in P, \quad x_{ik} \in \{0,1\}$

Objectif : la durée de travail du binôme k est $\sum_{i=1}^{n} c_i x_{ik}$. La date à. laquelle tous les projets sont terminés est donc $\max_{k \in \{1,\ldots,\frac{n}{2}\}} \sum_{i=1}^{n} c_i x_{ik}$. Il faut donc minimiser cette quantité :

$$\min(\max_{k \in \{1,\ldots,\frac{n}{2}\}} \sum_{i=1}^{n} c_i x_{ik})$$

Exercice 3 Sac à dos

Appliquez la méthode arborescente vue en cours pour résoudre le problème de sac à dos suivant (les objets sont déjà triés par w_i/p_i décroissants). On

précisera pour chaque noeud développé l'ensemble de solutions associées, son évaluation par défaut et son évaluation par excès (et comment elles sont calculées) ainsi que les opérations effectuées (séparation, troncature...).

$P = 14$

i	1	2	3	4	5	6	7
p_i	3	4	5	4	2	3	1
w_i	18	20	20	15	7	9	2

Chaque noeud S est associé à un ensemble $C(S)$ d'objets choisis et un ensemble $R(S)$ d'objets rejetés. Les solutions du noeuds S sont les sacs qui contiennent les objets de $C(S)$ et pas les objets de $R(S)$.

L'évaluation par excès $h(S)$ d'un noeud S est obtenue par l'algorithme qui consiste à trier les objets non encore dans $C(S) \cup R(S)$ par $\frac{w_i}{p_i}$ décroissant et à les placer un à un dans un sac déjà rempli par les objets de $C(S)$ jusqu'à ce que l'un des objets dépasse ou que le sac soit rempli exactement. En cas de dépassement on découpe l'objet en question de sorte qu'il remplisse le sac exactement. $h(S)$ est alors égal à la somme des profits des objets mis dans le sac par l'algorithme plus la fraction de produit qui correspond à la fraction du dernier objet.

L'évaluation par défaut $g(S)$ s'obtient en construisant une solution particulière du noeud S. Par exemple, on peut considérer l'algorithme qui calcule l'évaluation par excès, et simplement retirer de la solution construite l'objet découpé. $g(S)$ est alors égal au profit de cette solution. On peut améliorer l'évaluation en s'autorisant à remplir le sac avec des objets plus petits non encore examinés (mais nous ne le ferons pas dans le cadre de ce corrigé).

La séparation d'un noeud S se fait en choisissant un objet i qui n'appartient pas à $C(S)$ ni à $R(S)$ et à construire deux fils du noeud S, S' et S''. Par définition, $C(S') = C(S) \cup \{i\}, R(S') = R(S)$, et $C(S'') = C(S), R(S'') = R(S) \cup \{i\}$. Dans le cas présent, on choisira au noeud S l'objet i qui était découpé dans le calcul de l'évaluation par excès.

Le parcours de l'arborescence sera réalisé en utilisant la meilleure évaluation par excès d'abord.

Voici l'application de cette méthode à l'exemple. Au noeud racine S_0 on a $C(S_0) = R(S_0) = \emptyset$.L'algorithme pour le calcul de l'évaluation par excès place les objets $1, 2, 3$ dans le sac, ainsi que la moitié de l'objet 4. On obtient donc $h(S_0) = 65,5$ qu'on peut ramener à $h(S_0) = 65$ puisque le profit recherché est un entier. Par conséquent l'évaluation par défaut $g(S_0) = 58$. La meilleure solution rencontrée X_0 est initialisée au sac constitué des objets $1, 2, 3$.

Le noeud S_0 est ensuite séparé, en S_1 (sacs contenant l'objet 4) et S_2 (sacs ne contenant pas l'objet 4).

L'évaluation par excès du noeud S_1 est calculée en mettant dans le sac l'objet 4, puis les objets 1,2 et 3/5 de l'objet 3. D'où $h(S_1) = 65$, $g(S_1) = 53$ qui n'améliore pas la meilleure solution rencontrée.

L'évaluation par excès du noeud S_2 est calculée en mettant dans le sac les objets 1,2 3 et 5. On constate que le sac est rempli sans avoir à découper d'objets. D'où $h(S_2) = g(S_2) = 65$. X_0 est mis à jour et la meilleure solution rencontrée est maintenant le sac $1, 2, 3, 5$, et $f(X_0) = 65$ S_2 est fermé puisque son évaluation par excès égale son évaluation par défaut.

Considérons maintenant S_1. Comme $h(S_1) \leq f(X_0)$ il n'y aura pas de sacs de profit meilleur que X_0 dans le noeud S_1. Par conséquent il est fermé.

Toutes les feuilles étant fermées, la méthode arborescente est terminée, et le sac $1, 2, 3, 5$ est le sac optimal.

NB : si on utilise comme évaluation par défaut un algorithme qui cherche à remplir le sac en sautant simplement les objets qui dépassent, la méthode arborescente se termine en un seul noeud.

Exercice 4 Sac à dos inversé

On considère le problème suivant : n objets, de poids respectifs p_i et de regret r_i. On se donne également un entier P. On appelle regret d'un sac la somme des regrets des objets **qui ne sont pas** dans le sac. On cherche à construire un sac de poids inférieur ou égal à P qui soit de regret minimal.

1 Modéliser ce problème à l'aide d'un programme mathématique.

En définissant,pour tout objet i, x_i qui vaut 1 si l'objet est dans le sac et 0 sinon, on obtient les mêmes contraintes que le problème de sac à dos ($\forall i, \quad x_i \in \{0, 1\}$ et $\sum_{i=1}^{n} p_i x_i \leq P$). Par contre l'objectif change :

$$\min \sum_{i=1}^{n} r_i(1 - x_i)$$

Puisque l'objet i produit un regret s'il n'est pas dans le sac, c'est à dire si $1 - x_i = 1$.

2 On cherche à définir un schéma de programmation dynamique pour ce problème. Pour cela on définit la phase i comme étant celle du choix pour l'objet i, et l'état du système comme le poids des objets déjà dans le sac. Indiquer pour un couple (i, E) les décisions possibles à partir de cet état à la phase i, leur cout immédiat et les états résultants de ces décisions pour

la phase suivante.

A la phase i dans l'état E on peut prendre soit deux décisions (mettre l'objet i dans le sac δ_i^1 ou ne pas le mettre δ_i^0) soit une seule (δ_i^0 si i ne rentre pas, c'est à dire si $E + p_i \leq P$).

l'état résultant de la décision δ_i^0 est $(i+1, E)$. Celui résultant de la décision δ_i^1 est $(i+1, E+p_i)$ puisque l'état représente le poids des objets déjà dans le sac.

Le coût immédiat de δ_i^0 est r_i, celui de δ_i^1 est 0.

3 Notons $F_k(E)$ le regret minimum d'un sac de poids inférieur ou égal $P-E$ parmi les objets $\{k, \ldots, n\}$. Comment se définit la solution du problème initial par rapport à cette notation ?

$F_1(0)$ est le regret minimum d'un sac de poids inférieur ou égal à P parmi l'ensemble des objets $\{1, \ldots, n\}$.

4 Que vaut $F_n(E)$?

$$F_n(E) = \begin{cases} r_n & si \quad E + p_n > P \\ 0 & sinon \end{cases}$$

5 Etablir l'équation de récurrence satisfaite par les F_k.

$$F_k(E) = \begin{cases} r_k + F_{k+1}(E) & si \quad E + p_k > P \\ \min(r_k + F_{k+1}(E), F_{k+1}(E + p_k)) & sinon \end{cases}$$

6 En déduire un algorithme pour résoudre le problème dont vous donnerez la complexité.

L'algorithme consiste à calculer dans un premier temps $F_n(E)$ selon la formule de la question 4 pour E variant de 0 à P.

Puis dans une boucle qui fait varier k de $n-1$ à 1, on établit une boucle qui fait varier E de 0 à P à l'intérieur de laquelle on calcule $F_k(E)$ selon la formule de la question 5.

Pour finir, on reconstitue la solution optimale en partant de $E = 0$,et en faisant évoluer k de 1 à $n-1$. Si $F_k(E) = Fk + 1(E + p_k)$ on pose $x_k = 1$, et $E = E + p_k$, sinon, on pose $x_k = 0$.

Si après cette boucle $F_n(E) = 0$ alors $x_n = 1$ sinon $x_n = 1$.

La complexité de l'algorithme est en $O(nP)$.

7 Appliquez l'algorithme au problème suivant :

$P = 8$

i	1	2	3	4	5
p_i	3	4	5	4	2
r_i	15	18	20	12	6

$E\|i$	1	2	3	4	5
0	36	26	12	0	0
1	38	30	12	0	0
2	47	32	18	0	0
3	50	36	18	6	0
4	53	38	26	6	0
5	56	50	32	12	0
6	65	50	32	12	0
7	71	56	38	18	6
8	71	56	38	18	6

La solution optimale est de valeur 36 et correspond au sac contenant l'objet 1 et 3.