Examen terminal d'optimisation combinatoire
Durée 2h, le 19 juin 2007
Le barème est indicatif.

Exercice 1 Sac à dos dual(9 points)

On cherche à construire une méthode arborescente pour le problème suivant : comme pour le sac à dos, on a n objets, chacun ayant un poids p_i et une utilité w_i.On se donne également un entier W. On souhaite construire un sac d'utilité au moins égale à W qui soit de poids minimum.

On supposera que

$$\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}$$

1 Quelle est la solution optimale de ce problème lorsque tous les objets sont d'utilité 1 ?

Dans ce cas il suffit de les trier par poids croissant et de prendre les W premiers

2 Considérons le problème avec des utilités quelconques où l'on s'autorise à découper les objets en tranches d'utilité 1. Soit k le premier indice tel que $\sum_{i=1}^{k-1} w_i < W \leq \sum_{i=1}^{k} w_i$.

Montrer que

$$\alpha = \sum_{i=1}^{k-1} p_i + \frac{W - \sum_{i=1}^{k-1} w_i}{w_k} p_k$$

est le poids minimum d'un sac pour ce problème. **Chaque objet i est composé de w_i tranches de poids $\frac{p_i}{w_i}$. D'après la question précédente, la solution optimale consiste alors à prendre parmi ces tranches les W de plus petits poids. On peut remarquer que selon la numérotation choisie, toutes les tranches de l'objet 1 ont un poids inférieur à celles de l'objet 2, etc. Par conséquent, les W premières tranches comprennent toutes les tranches des objets jusqu'à $k-1$, et les $W - \sum_{i=1}^{k-1} w_i$ tranches restantes de l'objet k**

3 Calculer α pour l'exemple suivant :

$W = 40$

i	1	2	3	4	5	6
w_i	18	5	10	8	12	18
p_i	3	1	2	2	4	6

$k = 4$ **et donc** $\alpha = 3 + 1 + 2 + \frac{7}{4}$

4 En déduire que $\lceil \alpha \rceil$ est un minorant du poids optimal du problème initial (sans que les objets ne puissent être découpés). **Toute solution où les objets ne sont pas découpés est aussi une solution réalisable du problème où les objets sont découpés en tranches ; Par conséquent, le problème à objets découpés est une relaxation du problème initial. On en déduit, puisque l'on minimise, que la valeur optimale sur le problème relâché est inférieure ou égale à la valeur optimale du problème initial. Ainsi,** α **est un minorant du poids optimal. En observant que ce poids est nécessairement un entier, puisque les poids des objets le sont, on en déduit que la partie entière supérieure de** α **est bien un minorant du poids optimal.**

On définit une méthode arborescente pour ce problème. Chaque noeud S de l'arborescence est défini par deux ensembles $C(S)$ l'ensemble des objets choisis, et $R(S)$ l'ensemble des objets rejetés. Les solutions associées au noeud S sont les sacs d'utilité $\geq W$ qui contiennent tous les objets de $C(S)$ et aucun objet de $R(S)$.

5 Utiliser les questions précédentes pour proposer une évaluation par défaut $g(S)$ d'un noeud S **pour constituer une évaluation par défaut au noeud** S**, on note** $pc(S)$ **la somme des poids des objets choisis, et** $wc(S)$ **leur utilité. On considère le problème de sac posé sur les objets qui ne sont ni dans** $C(S)$ **ni dans** $R(S)$ **pour trouver la meilleure solution du noeud** S **: il s'agit d'un problème de même nature que le problème initial, avec moins d'objets et un objectif de profit de** $W - wc(S)$**. On peut donc, d'après la question 2 trouver la valeur** α^S **correspondant à ce problème. Si les objets restants ne permettent pas de remplir l'objectif de profit,** $\alpha^S = +\infty$**. On pose alors** $g(S) = \lceil \alpha^S \rceil + pc(S)$**.**

6 Utiliser les questions précédentes pour proposer une évaluation par excès $h(S)$ d'un noeud S **Pour obtenir une évaluation par excès dans un problème de minimisation, il suffit de construire une solution réalisable du noeud - s'il en existe-. Lors du calcul de l'évaluation par défaut, on met dans le sac un certain nombre d'objets (en plus de ceux de** $C(S)$ **) dont un est découpé. Il suffit de mettre entièrement cet objet dans le sac pour atteindre l'objectif de profit et avoir une solution réalisable**

7 Appliquez la méthode à l'exemple numérique de la question 3. On précisera

les évaluations de chaque noeud, le choix de l'objet de séparation, et le choix d'un parcours de l'arborescence. **A la racine, on a donc comme évaluation par défaut 8, et la solution réalisable construite consiste à mettre les objets 1,2,3 et 4 en entier dans le sac pour un poids 8 également. Comme l'évaluation par défaut égale l'évaluation par excès, il s'agit de la solution optimale.**

Exercice 2 Consommation d'énergie(5 points + bonus)

On se donne une suite de n tâches à effectuer sur une machine. Cette machine possède deux modes de fonctionnement différents. Lorsqu'une tâche i est effectuée selon le mode 1, elle a une durée a_i, et selon le mode 2 elle dure b_i. Le choix d'un mode pour effectuer une tâche i engendre un coût énergétique : le choix du mode 1 coûte $\frac{c_1}{a_i}$, et le choix du mode 2 $\frac{c_2}{b_i}$.

On cherche à associer à chaque tâche un mode de la machine selon lequel elle sera exécutée, de sorte que la durée totale de l'ordonnancement des tâches (on supposera que l'on fait les tâches en séquence, une fois qu'on a décidé du mode de chacune) ne dépasse pas une valeur donnée D, afin que la dépense totale d'énergie soit minimale. S'il n'existe aucune solution réalisable, on dira par convention que la dépense minimale d'énergie vaut $+\infty$.

1 Modéliser ce problème à l'aide d'un programme mathématique. **En prenant une variable bivalente x_i pour chaque tâche i, qui vaut 0 lorsque la machine l'exécute en mode 1 et 1 lorsqu'elle l'exécute en mode 2, on a le problème suivant :**

$$\begin{cases} \min \sum_{i=1}^{n} \frac{c_1}{a_i}(1-x_i) & + \sum_{i=1}^{n} \frac{c_2}{b_i} x_i \\ \sum_{i=1}^{n} a_i(1-x_i) & + \sum_{i=1}^{n} b_i x_i \leq D \\ \forall i, & x_i \in \{0,1\} \end{cases}$$

On souhaite décrire un schéma de programmation dynamique pour résoudre ce problème. Soit $F_k(E)$ le coût minimal en énergie d'un ordonnancement de durée inférieure ou égale à E pour les tâches $\{k, \ldots, n\}$.

2 Comment calculer $F_n(E)$? $F_n(E)$ **vaut $+\infty$ si la tâche i ne peut pas être faite en temps voulu - si $E < min(a_n, b_n)$. Si $a_n \leq E < b_n$ alors $F_n(E) = \frac{c_1}{a_n}$. Si $b_n \leq E < a_n$ alors $F_n(E) = \frac{c_2}{b_n}$. Sinon,** $F_n(E) =$

$\min(\frac{c_1}{a_n}, \frac{c_2}{b_n})$

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

$$F_k(E) = \begin{cases} +\infty & si \quad E < min(a_k, b_k) \\ \frac{c_1}{a_k} + F_{k+1}(E - a_k) & si \quad a_k \leq E < b_k \\ \frac{c_2}{b_k} + F_{k+1}(E - b_k) & si \quad b_k \leq E < a_k \\ \min(\frac{c_1}{a_k} + F_{k+1}(E - a_k), \frac{c_2}{b_k} + F_{k+1}(E - b_k)) & sinon \end{cases}$$

4 En déduire un algorithme de programmation dynamique qui construit une solution optimale dont on précisera la complexité.

```
for E=0 to D \\
calcul de Fn(E)\\
For k=n-1 to 1\\
   for E=0 to D\\
   calcul de Fk(E) selon formule de récurrence\\
la valeur recherchée est F1(D).
```

pour obtenir la solution optimale, il faut refaire une boucle, en partant de $k = 1$ et de $E = D$. A chaque itération, si $F_k(E) = \frac{c_1}{a_k} + F_{k+1}(E - a_k)$ alors on pose $E = E - a_k$ et $x_k = 0$, si $F_k(E) = \frac{c_2}{b_k} + F_{k+1}(E - b_k)$ alors on pose $E = E - b_k$ et $x_k = 1$.

La complexité de l'ensemble est en $O(nD)$.

5 (facultative - 2 points) Appliquez cet algorithme aux données suivantes :

Tâches :

i	1	2	3	4	5	6
a_i	3	1	2	2	4	3
b_i	2	1	1	1	3	2

machine : $c_1 = 1, c_2 = 2, D = 12$

Exercice 3 Recouvrement de sommets (6 points)

Etant donné un graphe non orienté $G = (S, E)$, un recouvrement est un sous-ensemble des sommets X de sorte que pour toute arête $\{x, y\} \in E$, $x \in X$ ou $y \in X$. On s'intéresse à la recherche d'un recouvrement de cardinalité minimale. On notera X^* un recouvrement optimal.

1 Soit U un sous-ensemble d'arêtes du graphe qui n'ont aucune extrémité commune. Montrer que pour tout recouvrement X de G, $|X| \geq |U|$ (le nombre de sommets de X est au moins égal au nombre d'arêtes de U).

Chaque arête de U possède une extrémité dans X. Comme par hypothèse toute ces extrémités sont distinctes, il y a au moins $|U|$ sommets dans X.

Considérons l'algorithme suivant qui construit un ensemble de sommets X et un ensemble d'arêtes U :

$E' = E$
$X = \emptyset$
$U = \emptyset$
tant que $E' \neq \emptyset$

choisir $\{x, y\} \in E'$

ajouter x et y à X

ajouter l'arête $\{x, y\}$ à U

retirer de E' toutes les arêtes adjacentes à x et à y

2 Tracer un graphe à 6 sommets et 10 arêtes et appliquez cet algorithme à votre graphe.

3 Montrer que l'ensemble X construit par l'algorithme est bien un recouvrement de G, et que $|X| = 2 * |U|$ **Au cours de l'algorithme, une arête peut être soit mise dans U, soit éliminée. Lorsqu'une arête est dans U, ses deux extrémités sont dans X. Lorsqu'une arête est éliminée, c'est qu'elle est adjacente à une arête de U, et donc à un sommet de X. Donc X est un recouvrement dont le cardinal est exactement deux fois celui de U.**

4 Montrer qu'à la fin de l'algorithme, les arêtes de U n'ont aucune extrémité

commune **Si deux arêtes de U avaient une même extrémité, en se plaçant à l'itération de l'algorithme où la première a été mise dans U, on observerait que la seconde a nécessairement été éliminée, et donc ne peut pas être choisie dans U ultérieurement.**

5 En déduire que l'algorithme ci-dessus est un algorithme approché avec un rapport d'approximation relative égal à 2. **D'après la question précédente, les arêtes de U n'ont aucune extrémité commune. Par conséquent, d'après la question 1, le recouvrement optimal X^* possède au moins $|U|$ sommets. Comme X construit par l'algorithme en possède $2|U|$, on a $|X| = 2|U| \leq 2|X^*|$. L'algorithme construit donc un recouvrement qui contient au plus deux fois le nombre optimal de sommets. C'est un algorithme 2-approché.**