Majeure Informatique Promotion 2004

Cours de Conception et analyse d'algorithmes
Examen du mercredi 28 mars 2007. 2 heures
Documents autorisés : poly, transparents du cours, énoncés de PC

Le barême est indicatif. On vous demande d'accorder une attention particulière à une rédaction soignée, qui sera un critère important dans l'évaluation.

I. Problèmes NP-complets et algorithmes d'approximation *(7 points).*

L'objet de ce problème est d'étudier un problème d'ordonnancement très simple : il y a n travaux à réaliser sur m machines ; chaque travail A_i nécessite un temps p_i et doit être effectué sur une seule machine sans interruption. On cherche à déterminer une affectation des travaux sur les machines qui minimise le temps de la réalisation de l'ensemble des travaux (c'est à dire le temps de la machine qui travaille le plus).

Par exemple si on a 3 machines et 6 travaux de durées : $1, 6, 2, 5, 4, 7$ ce temps minimal est 9, il est obtenu en affectant les trois premiers travaux sur une même machine, puis les deux suivants sur une autre enfin le dernier sur la troisième.

1. Donner une version *décision* du problème et démontrer que celle-ci est NP-complète.

2. Exprimer ce problème sous la forme de l'optimisation, sous contraintes, d'une fonction à nm variables entières.

3. Soit T_{opt} le temps obtenu pour l'affectation optimale, montrer que :

$$mT_{opt} \geq \sum_{i=1}^{n} p_i$$

On considère l'algorithme glouton qui consiste d'abord à classer de façon quelconque tous les travaux pour former une liste d'attente ordonnée : $A_1, A_2, \ldots, A_n$; ensuite, à chaque fois qu'une machine est disponible on lui affecte le premier travail sur la liste d'attente. On remarque que cet algorithme donne un temps de 13 pour l'exemple proposé plus haut.

Soit A_k le travail qui se termine en dernier dans l'algoritme glouton et soit t_k l'heure du début de l'exécution de ce travail pour cet algorithme.

4. Montrer que :

$$t_k \quad \leq \quad \frac{1}{m} \sum_{i \neq k} p_i$$

En déduire une majoration du temps T_g obtenu par l'algorithme glouton.

5. En déduire que l'algorithme glouton est $\frac{m-1}{2m-1}$-approché.

II. Un algorithme probabiliste pour compter *(6 points).*

On considère le problème COMPTER(F) pour lequel la donnée est une formule booléenne F en forme disjonctive, c'est à dire qu'elle se présente sous la forme d'un *ou* de plusieurs monômes F_i, $(F = F_1 \vee F_2 \vee \cdots \vee F_p)$; où chaque F_i est le *et* de plusieurs variables (complémentées ou pas) $(F_i = y_1 \wedge y_2 \wedge \cdots \wedge y_{q_i})$.

Il s'agit alors de déterminer le nombre $c(F)$ de valeurs des variables $x_1, x_2, \ldots x_n$ qui donnent la valeur *true* à F.

1. Déterminer $c(F)$ pour la fonction à 4 variables :

$$F = (x_1 \wedge \overline{x}_3) \ \vee \ (\overline{x}_2 \wedge \overline{x}_3) \ \vee \ (x_1 \wedge x_2 \wedge x_4)$$

2. On applique une méthode aléatoire qui consiste à tirer au hasard avec probabilité $\frac{1}{2}$ la valeur (*true* ou $false$) de chacune des n variables et de déterminer si l'affectation trouvée donne la valeur *true* à F. Le tirage est effectué m fois et le nombre de fois où la valeur *true* est obtenue pour F est m_1, quelle est alors une approximation de $c(F)$?

L'expérience montre que l'approximation trouvée n'est pas bonne. On propose alors un nouvel algorithme qui utilise les nombres u_i d'affectations qui donnent la valeur *true* à F_i.

- On tire au hasard un nombre a entre 1 et $\sum_{i=1}^{p} u_i$, et on détermine j tel que

$$\sum_{i=1}^{j-1} u_i < a \leq \sum_{i=1}^{j} u_i$$

- On tire au hasard une affectation de valeurs qui donnent la valeur *true* à F_j et on cherche si cette affectation donne aussi la valeur *true* à un F_i tel que $i < j$; si tel n'est pas le cas on incrémente le compteur du nombre de tirages réussis.
- Après m tirages on obtient $m_1 \leq m$ tirages réussis, il s'agit alors d'obtenir une approximation de $c(F)$

3.

- **3.1** On note U_i l'ensemble des affectations qui donnent la valeur *true* à F_i, quelle est le nombre d'élements u_i de U_i si F_i contient q_i variables sur un total de n ?
- **3.2** On note U l'ensemble des couples (i, α) tels que i est un nombre compris entre 1 et p et α une affectation qui donne la valeur *true* à F_i. Quel est le cardinal de U en fonction des q_i de p et de n ?
- **3.3** Soit V le sous ensemble de U formé des (i, α) tels que α est une affectation des valeurs des variables qui donnent la valeur $false$ à tous les F_j pour $j < i$. Exprimer à l'aide de V la valeur de $c(F)$
- **3.4** Donner une approximation de $c(F)$ à l'aide des nombres m de tirages et m_1 de tirages réussis donnés par l'algorithme.
- **3.5** Cette méthode donne-t-elle à votre avis une meilleure approximation de $c(F)$ que celle de la question **2** ? Expliquer pourquoi.

III. Arbres recouvrants dynamiques *(7 points).*

Soit un graphe $G = (X, E, w)$ non orienté dont l'ensemble de sommets est $1, 2, \ldots, n$ et dont l'ensemble des arêtes $E = \{e_1, e_2, \ldots e_m\}$ est muni d'un poids w (une application de E dans l'ensemble des rationnels positifs) tel que tous les poids sont différents. On adopte la convention :

$$0 < w(e_1) < w(e_2) < \ldots < w(e_{m-1}) < w(e_m)$$

Dans ce cas il existe *un et un seul arbre recouvrant de poids minimum,* noté T que l'on supposera calculé.

On considère dans cet exercice la modification de la valeur du poids d'une des arêtes du graphe ; pour les deux premières questions cette arête n'appartiendra pas à T et pour les deux questions suivantes, ce sera un élément de T.

1. Donner un algorithme sans justification (celle-ci fera l'objet de la question suivante) qui calcule le nouvel arbre minimal lorsque l'on diminue d'une quantité $\alpha > 0$ le poids d'une seule arête $e_k = \{a, b\}$ qui n'est pas dans T. On supposera que l'arbre T est donné par une racine et un tableau `pere`, tel que pour chaque sommet i, `pere[i]` est le premier sommet rencontré en remontant de i à la racine. La complexité de votre algorithme devra être en $O(\ell)$ (où ℓ est la longueur du cycle de $T \cup \{e_k\}$).

2. Jusifier la validité de l'algorithme que vous avez proposé pour répondre à la question précédente. Pour cela vous considèrerez le nouveau poids w' ($w'(e) = w(e)$ si $e \neq e_k$ et $w'(e_k) = w(e_k) - \alpha$), l'indice j tel que $w(e_j) < w'(e_k) < w(e_{j+1})$ et $F_1, F_2, \ldots, F_p$ l'ensemble des composantes connexes de $T \cap \{e_1, e_2, \ldots, e_j\}$. Vous comparerez les deux exécutions de l'algorithme de Kruskal : celle avec le poids w et celle avec le poids w'.

3. On considère maintenant le cas où on augmente de $\alpha > 0$ le poids d'une arête $e_k = \{a, b\}$ de T, proposer un algorithme en $O(m)$ qui calcule le nouvel arbre de poids minimum.

4. Justifier la validité de votre algorithme par un raisonnement du même type que celui qui a été fait pour la Question 2.

5. L'hypothèse suivant laquelle tous les poids des arêtes de G sont différents, est-elle nécessaire pour la vailidité des algorithmes que vous avez proposés ?