

Cours de Conception et analyse d'algorithmes

Corrigé de l'examen du mercredi 26 mars 2008. 2 heures

Documents autorisés : poly, transparents du cours, énoncés de PC

I. Algorithme glouton (4 points).

Le problème de l'arbre couvrant de poids minimum a été traité en cours :

Données : Un graphe connexe $G = (X, E)$ et une valuation v des arêtes.

Problème : Trouver un arbre couvrant de poids minimum.

On étudie un algorithme différent de celui vu en cours pour résoudre ce problème.

1. Démontrer la propriété suivante :

- Soit C un cycle du graphe G et e l'arête de valuation maximale sur ce cycle. Alors il existe un arbre couvrant de poids minimum qui ne contient pas e .

Corrigé: Soit T un arbre couvrant de poids minimum. S'il contient l'arête e alors $T - e$ se décompose en 2 composantes connexes qui forment une partition (T_1, T_2) des sommets de G . Tout cycle contient un nombre pair d'arêtes de la coupe entre T_1 et T_2 . Comme e fait partie de cette coupe, il existe au moins une autre arête e' de C y appartenant et par hypothèse sur e et C , $v(e') \leq v(e)$. Comme $T - e + e'$ est connexe et contient le même nombre d'arêtes que T , c'est un arbre, de poids inférieur ou égal au poids de T donc minimum. \square

L'algorithme que l'on considère est le suivant :

- Classer les arêtes par valuations décroissantes, et poser $I = E$.
- Pour chaque arête dans l'ordre des valuations décroissantes,
 - si e appartient à un cycle de I supprimer e de I .
- renvoyer l'ensemble I des arêtes restantes.

2. Montrer que cet algorithme est correct.

Corrigé: L'algorithme ne supprime que des arêtes appartenant à un cycle : le graphe restant est donc toujours connexe. De plus il ne contient plus de cycle, donc c'est un arbre, qui est couvrant.

Soit V le poids d'un arbre couvrant de poids minimum de G . Montrons qu'à tout moment lors de l'exécution de l'algorithme, l'arbre couvrant de poids minimum du graphe $G_I = (X, I)$ est de poids V : on saura ainsi que l'arbre renvoyé par l'algorithme est de poids V .

L'ensemble I ne change que lorsqu'on considère une arête e appartenant à un cycle de G_I : on sait alors qu'il existe un arbre couvrant de poids minimum de G_I ne contenant pas e , puisque e est maximale dans son cycle (à cause de l'ordre des valuations décroissantes). Cet arbre est donc aussi un arbre couvrant de poids minimum de G_{I-e} , et la propriété reste vraie. \square

3. À chaque étape de l'algorithme il faut vérifier si e appartient à un cycle de I . Proposer une méthode pour faire cette vérification en temps linéaire.

Corrigé: On effectue un parcours en profondeur du graphe obtenu en supprimant l'arête $e = \{x, y\}$ à partir du sommet x . L'arête e appartient à un cycle si et seulement si le parcours visite le sommet y . La complexité est linéaire en le nombre d'arêtes. \square

4. Donner la complexité globale de l'algorithme. Est-il compétitif avec l'algorithme vu en cours ?

Corrigé: L'algorithme est de complexité $O(m^2)$ où m est le nombre d'arêtes de G , ce qui n'est pas compétitif avec l'algorithme de Kruskal (en $m \log n$). \square

II. Programmation dynamique (4 points).

On se donne un texte de longueur n dont on pense qu'il a été obtenu à partir d'un document en français en supprimant tous les espaces et symboles de ponctuation ("long-tempsjemesuislevédebonneheure..."). On souhaite reconstruire le document à l'aide d'un dictionnaire, auquel on accède par une fonction booléenne `dict()` qui renvoie pour toute chaîne s

$$\text{dict}(s) = \begin{cases} \text{vrai} & \text{si } w \text{ est un mot du dictionnaire} \\ \text{faux} & \text{sinon.} \end{cases}$$

1. Décrire un algorithme qui décide si le texte peut être décomposé en mots du dictionnaire. L'algorithme devrait être de complexité $O(n^2)$ (un appel à la fonction `dict` étant compté comme une opération élémentaire).

Corrigé: On note $s[i, j]$ la partie du texte allant de la position i à la position j (incluse), de sorte que $s = s[1, n]$.

On considère la table booléenne $t(i)$ qui dit si le facteur gauche $s[1, i]$ de longueur i du texte peut être décomposé : cette table vérifie $t(0) = \text{VRAI}$ et satisfait la récurrence

$$t(i) = \bigvee_{1 \leq j \leq i} (\text{dict}(s[j, i]) \wedge t(j - 1)).$$

Le remplissage de cette table prend donc un temps $O(n^2)$, sauf si on a une borne M sur la taille du plus long mot du dictionnaire, auquel cas la complexité est $O(Mn)$. \square

2. Dans le cas où le texte peut être décomposé, donner un algorithme qui propose une reconstitution possible. Quelle est la complexité de votre algorithme, et l'espace mémoire dont il a besoin ?

Corrigé: Lors de la construction de la table $t(i)$ on remplit en parallèle une table $p(i)$ qui indique pour chaque position i telle que $t(i)$ est VRAI un indice j tel que $\text{dict}(s[j, i]) \wedge t(j - 1)$. Il suffit alors de suivre les indices en partant de $p(n)$ pour reconstruire une décomposition. L'espace utilisé est linéaire ainsi que le temps de calcul une fois les tables constituées. \square

III. Programmation linéaire (4 points)

Étant donné un système S d'inéquations

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &\leq b_1 \\ &\vdots \\ a_{m1}x_1 + \dots + a_{mn}x_n &\leq b_m \end{aligned}$$

on dit que la j ème inéquation est forcée à l'égalité si toute solution x du système est telle que $a_{j1}x_1 + \dots + a_{jn}x_n = b_j$. Soit F_S l'ensemble des indices des inéquations forcées à l'égalité de S .

1. Montrer qu'il existe une solution x^* de S telle que :

$$a_{j1}x_1^* + \dots + a_{jn}x_n^* = b_j \text{ si et seulement si } j \in F.$$

(Indication : l'ensemble des solutions de S est convexe.)

Corrigé: Pour tout $j \notin F$ il existe une solution x^j telle que $a_{j1}x_1^j + \dots + a_{jn}x_n^j < b_j$. Soit $k = m - |F|$. Le barycentre $x^* = \frac{1}{k} \sum_{j \notin F} x^j$ est un point admissible par convexité et il satisfait la condition requise. \square

2. Donner un algorithme, basé sur la programmation linéaire, qui permet de déterminer quelles sont les équations forcées à l'égalité dans le système S et qui donne une solution x^* satisfaisant la condition de la question précédente.

(Indication : il pourra être utile de résoudre plusieurs problèmes de programmation linéaire pour arriver au résultat.)

Corrigé: Pour chaque $j \in \{1, \dots, m\}$ on considère le problème de programmation linéaire donné par les contraintes S et la fonction d'objectif $\min_x a_{j1}x_1 + \dots + a_{jn}x_n$. Si la solution x^j renvoyée vérifie l'égalité l'équation est forcée à l'égalité, sinon on la garde pour obtenir à la fin x^* en prenant un barycentre. \square

IV. NP-Complétude (4 points).

Utiliser la NP-complétude du problème STABLE pour montrer que le problème RECOUVREMENTEXACT ci-dessous est lui aussi NP-complet :

Données. Une matrice A de taille $m \times n$ à coefficient 0, 1.

Problème RECOUVREMENTEXACT. Décider s'il existe un vecteur x à coefficient 0, 1 tel que $Ax = \mathbf{1}$, où $\mathbf{1}$ est le vecteur avec m coordonnées égales à 1.

Indications : • Il existe un stable de taille k dans le graphe G si et seulement si il existe une orientation des arêtes de G telle qu'au moins k sommets n'ont pas d'arêtes sortantes.

• Si la matrice extraite aux lignes $I = \{i, i+1, i+2\}$ et colonnes $J = \{j, j+1, j+2, k, \ell\}$ est

1	1	0	0	0	0
1	0	1	0	1	0
0	1	0	1	0	1

et si la ligne i ne contient pas d'autre entrée non nulle, alors toute solution x du problème satisfait $x_k \cdot x_\ell = 0$.

Corrigé: Étant donné une instance (G, k) du problème STABLE, où G à n sommets et m arêtes. On construit une matrice A avec $3m + n + (n - k)$ lignes et $5m + n + n(n - k)$ colonnes :

- Les $3m$ premières lignes et colonnes sont associées aux arêtes : à l'arête i ($i = 0, \dots, m - 1$) on associe la matrice extraite de A correspondant aux lignes $\{3i + 1, 3i + 2, 3i + 3\}$ et colonnes $\{5i + 1, 5i + 2, 5i + 3, 5i + 4, 5i + 5\}$ qu'on remplit avec les 5 premières colonnes de la matrice de l'indication. Les autres entrées de ces colonnes sont nulles ainsi que celles des lignes de la forme $3i + 1$.
- À chaque sommet j ($j = 1, \dots, n$) on associe la ligne et la colonne $3m + j$: on met un 1 dans la case $(3m + j, 5m + j)$ et ainsi que dans les cases $(3i + 2, 5m + j)$ pour les arêtes i dont j est l'origine et $(3i + 3, 5m + j)$ pour les arêtes i dont j est l'extrémité.
- Enfin les $n(n - k)$ dernières colonnes servent de cache pour les sommets non sélectionnés : la colonne $4m + n + pq$ contient un 1 en position $3m + p$ et un 1 en position $3m + n + q$.

Comme exactement $n - k$ des $n(n - k)$ dernières colonnes doivent apparaître, k sommets doivent être sélectionnés, ce qui n'est possible que s'il existe une affectation des arêtes à k sommets. \square

V. Algorithme d'approximation et limite d'approximabilité (4 points).

On se propose d'étudier le problème k -REGROUPEMENT EUCLIDIEN suivant :

Données. Un entier k et un ensemble $X = \{x_1, \dots, x_n\}$ de points et leurs distances $d(x_i, x_j)$ pour tous i, j . La distance satisfait l'inégalité triangulaire : $d(x, y) \leq d(x, z) + d(z, y)$ pour tout triplet (x, y, z) .

Problème. Trouver une partition des points en k groupes G_1, \dots, G_k qui minimise la quantité

$$\max_{\ell} \min_{x_i, x_j \in G_{\ell}} d(x_i, x_j)$$

qui donne le diamètre du groupe de plus grand diamètre.

On souhaite analyser l'algorithme d'approximation suivant :

- Soit y_1 un point quelconque de X .
- Pour i de 2 à k choisir pour y_i le point le plus éloigné des y_j déjà choisis.
- Former k groupes associés aux y_i en affectant chaque point x au groupe tel que $d(x, y_i)$ soit minimum.

1. Soit y_{k+1} le point le plus éloigné de $\{y_1, \dots, y_k\}$ et $r = \inf_{1 \leq i \leq k} (d(y_{k+1}, y_i))$ la distance de y_{k+1} à cet ensemble de points. Montrer que le diamètre de chaque groupe est inférieur à $2r$.

2. Montrer que l'algorithme proposé fournit une solution à un facteur 2 de l'optimum.

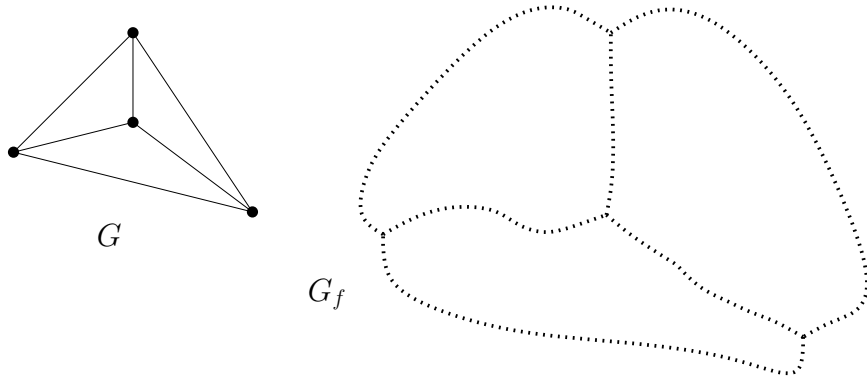
La NP-complétude du problème RECOUVREMENT-PAR-SOMMETS (ou COUVRANT) de couverture des arêtes d'un graphe par k sommets a été vue en cours. On admet que ce

problème reste NP-complet pour les graphes planaires *cubiques* (dont tous les sommets sont de degré 3).

À un graphe cubique $G = (X, E)$ et d'une fonction $f : E \rightarrow \mathbb{N}^*$ on associe le graphe subdivisé G_f obtenu en remplaçant chaque arête de G par une chaîne de $2f(e) + 1$ arêtes (autrement dit, en subdivisant chaque arête e à l'aide de $2f(e)$ sommets de degré 2).

On admet que pour tout graphe planaire cubique G il existe une fonction de subdivision f telle qu'il soit possible de dessiner en temps polynomial G_f dans le plan de sorte que :

- le dessin soit planaire (pas de croisement d'arêtes ou de sommets identifiés) ;
- les arêtes soient représentées par des segments de longueur unité ;
- deux sommets non adjacents dans le graphe G_f sont à distance au moins $\sqrt{3}$ l'un de l'autre dans le plan (en particulier l'angle formé par 2 arêtes adjacentes autour d'un sommet est au moins $2\pi/3$, exactement pour les sommets cubiques).



On note $M(G_f)$ l'ensemble des milieux des arêtes de G_f .

3. Montrer qu'il existe un ensemble de sommets recouvrants de G avec k sommets si et seulement si il existe un regroupement des points de $M(G_f)$ en $k + \sum_e f(e)$ groupes de diamètre au plus 1.

4. Montrer qu'un groupe de 4 points de $M(G_f)$ contient toujours deux points à distance au moins $d_0 = (1 + \sqrt{3})/\sqrt{2}$ et qu'un groupe de 3 points à distance inférieure à d_0 les uns des autres est nécessairement formé des milieux de 3 arêtes incidentes à un sommet de degré 3.

5. Dédire de la discussion précédente si on peut approcher le problème k -REGROUPEMENT EUCLIDIEN à un facteur d_0 près alors $\mathcal{P} = \mathcal{NP}$.