

**Cours de *conception et analyse d'algorithmes***

TD 9 – Recherche locale et révision

30 novembre 2011

**1. Comprendre en détail l'algo de classification par recherche locale du cours.**

On rappelle le problème : étant donné un entier  $k$  une instance du problème  $k$ -CLASSIFY est spécifié par un graphe  $G$  dont les arêtes portent des poids  $p_{xy}$  et les sommets des vecteurs  $v_x = (v_x^1, \dots, v_x^k)$ ; il s'agit de trouver un étiquetage des sommets  $\ell : V \rightarrow \{1, \dots, k\}$  qui minimise la quantité

$$\Phi(\ell) = \sum_{x \in V} v_x^{\ell(x)} \sum_{xy \in E} \delta_{\ell(x) \neq \ell(y)} p_{xy}.$$

1. Écrire proprement la réduction du cas  $k = 2$  à un problème de coupe minimale
2. L'algo de recherche locale donné en cours se base sur la notion de voisinage suivante : les voisins d'un étiquetage  $\ell$  sont les étiquetages obtenus en choisissant une couleur  $i$  et un ensemble de sommets  $X \subset V$  et en posant  $\ell_{X,i}(x) = i$  si  $x \in X$ ,  $\ell_{X,i}(x) = \ell(x)$  sinon. Écrire là encore la réduction de la recherche d'un voisin améliorant à un problème de coupe.
3. Vérifier qu'un optimum local obtenu par l'algo précédent approche l'optimum global à un facteur 2.

**2. Équilibrage de charge**

On considère  $m$  machines identiques et une séquence de  $n$  tâches auxquelles sont associés des charges de travail  $p_1, \dots, p_n$ . Le but est de répartir les tâches de manière à équilibrer les charges, ou autrement dit, à minimiser le maximum des sommes des charges des machines.

**i.** Dans un premier temps on suppose que les tâches sont distribuées arbitrairement sur les machines au départ et que l'équilibrage se fait localement : à chaque étape, deux machines sont mises en contact et échantent des tâches de manière à équilibrer leur charge. Autrement dit un échange de tâches est favorable si le maximum des deux charges après l'échange est strictement inférieur au maximum des deux charges avant l'échange.

Une distribution de charge est stable si aucun échange favorable n'est possible.

**a.** Montrer qu'il existe toujours une distribution de charge stable et qu'on peut l'obtenir par recherche locale.

**b.** Montrer que toute distribution de charge stable a une charge maximale inférieure à deux fois l'optimum.

**ii.** On considère maintenant la variante online : il faut placer les tâches sur les machines au fur et à mesure de leur arrivée en minimisant la charge maximale. (Les tâches ne finissent pas durant l'exercice.)

Montrer que l'algo glouton qui place chaque tâches sur la machine la moins chargée est  $\alpha$ -compétitif pour  $\alpha = 2$  ou mieux encore pour  $\alpha = 2 - 1/m$ .

### 3. Arbre d'évolution optimal pour la distance de Hamming.

On dispose d'un certain nombre de séquences génétiques et on souhaite expliquer l'évolution de ces séquences par un arbre tel que deux séquences voisines dans l'arbre sont à faible distance. Pour cela on considère la distance de Hamming  $d_H$  entre mots.

Un arbre d'évolution pour les séquences  $f_1, f_2, \dots, f_m$  est un arbre dont les sommets sont  $f_1, f_2, \dots, f_m$ , son coût est la somme des  $m - 1$  distances entre les séquences reliées par une arête

i. Donner un algorithme en temps polynomial qui résout le calcul de l'arbre d'évolution de coût minimal.

On complique un peu le problème en supposant donnés deux ensembles de séquences  $F = \{f_1, f_2, \dots, f_m\}$  et  $G = \{g_1, g_2, \dots, g_n\}$  on veut expliquer l'évolution de tous les éléments de  $F$  en s'aidant d'une partie de  $G$ . Pour cela on cherche un sous-ensemble  $G'$  de  $G$  et un arbre d'évolution dont l'ensemble des sommets est  $F \cup G'$  dont le coût est minimal.

On se propose de montrer que le problème de décision associé est alors  $\mathcal{NP}$ -complet. Pour cela on réduit RECOUVREMENT-PAR-SOMMETS à notre problème. Soit  $\Gamma = (X, E)$  un graphe connexe ayant  $n$  sommets  $x_1, x_2, \dots, x_n$  et  $m$  arêtes. On associe à  $\Gamma$  un ensemble  $F$  et un ensemble  $G$  de mots de longueur  $n$  sur l'alphabet  $\{0, 1\}$  définis comme suit. On note  $w_\emptyset$  le mot fait de  $n$  0, on note  $w_i$  pour  $1 \leq i \leq n$  le mot fait de  $n - 1$  0 et un 1 en position  $i$  et on note  $w_{i,j}$  pour  $1 \leq i < j \leq n$  le mot fait de  $n - 2$  0 et deux 1 en positions  $i$  et  $j$ . On définit alors l'ensemble  $G = \{w_1, w_2, \dots, w_n\}$  dont les éléments correspondent aux sommets de  $\Gamma$  et l'ensemble  $F = \{w_\emptyset\} \cup \{w_{i,j} / 0 \leq i < j \leq n \text{ et } \{x_i, x_j\} \in E\}$  dont les éléments différents de  $w_\emptyset$  correspondent aux arêtes de  $\Gamma$ .

ii. Utiliser cette construction pour montrer que la recherche d'un arbre d'évolution conditionné de coût inférieur à  $k$  est un problème  $\mathcal{NP}$ -complet.

### 4. Placement optimal de ressources

L'objet de ce problème est d'étudier un problème de placement optimal de ressources : on souhaite placer  $k$  supermarchés dans  $n$  villes de façon à minimiser la distance maximale des villes au supermarché le plus proche.

Soit  $G = (X, E)$  un graphe non orienté. Un *ensemble dominant* de  $G$  est un sous-ensemble  $S \subset X$  tel que tout sommet de  $X \setminus S$  est voisin d'un sommet de  $S$ .

i. Donner une version décision du problème DOMINANT suivant : étant donné un graphe  $G$ , trouver la cardinalité minimale d'un ensemble dominant de  $G$ .

ii. Montrer que la version décision de DOMINANT est NP-complet (on pourra utiliser une réduction à partir de RECOUVREMENT D'ENSEMBLES).

Pour toutes villes  $x$  et  $y$ , la distance entre ces deux villes est notée  $d(x, y)$ . On suppose que la fonction  $d$  satisfait l'inégalité triangulaire : ie.  $d(x, y) \leq d(x, z) + d(z, y)$  pour tout triplet  $(x, y, z)$  de villes. Le problème  $k$ -CENTRES consiste à trouver, dans un graphe complet à  $n$  sommets dont les arêtes sont valuées par une fonction  $d$  satisfaisant l'inégalité triangulaire, un sous-ensemble  $S$  de  $k$  sommets minimisant la quantité :

$$\phi(S) = \max_{y \notin S} \min_{x \in S} d(x, y).$$

On note  $\phi^*$  le minimum  $\min_{S \subset V} \phi(S)$ .

Étant donné un graphe non-orienté  $G = (X, E)$ , on considère le graphe complet  $K_X$  sur  $X$  et une valuation des arêtes définie par  $d(x, y) = 1$  si  $(x, y) \in E$  et  $d(x, y) = 2$  sinon.

iii. Utiliser cette construction et le problème DOMINANT pour montrer que la version du problème  $k$ -CENTRES est NP-complète.

iv. En déduire que s'il existe un algorithme polynomial qui trouve, étant donné un entier  $k$  et une valuation  $d$  du graphe complet à  $n$  sommets, un ensemble  $S$  de taille  $k$  tel que  $\phi(S) < 2\phi^*$  alors  $P = NP$ .

## 5. Programmation dynamique : Alignement de séquences

On considère des séquences sur un alphabet  $\Sigma$ . On se donne une fonction  $\delta : (\Sigma \cup \{-\})^2 \mapsto \mathbb{N}$  qui représente le *score d'alignement* entre lettres. On pourra supposer que  $\delta(x, x) > 0$  pour tout  $x$  appartenant à  $\Sigma$ ,  $\delta(x, y) = \delta(y, x) < 0$  pour  $x \neq y$  appartenant à  $\Sigma \cup \{-\}$  et  $\delta(-, -) < 0$ . Dans le cours sur la programmation dynamique, on a vu qu'on pouvait obtenir le meilleur alignement pour deux séquences en espace linéaire. La technique utilisée mixait programmation dynamique et approche "diviser pour régner". On étudie ici des variantes de ce problème.

i. Proposer un algorithme en  $O(n^3)$  qui étant donné trois séquences  $u, v, w$  de longueur  $n$ , détermine l'alignement réalisant le meilleur score.

ii. Proposer un algorithme en  $O(n^2)$  qui étant donné deux séquences  $u$  et  $v$  de longueur  $n$ , détermine les facteurs  $u'$  et  $v'$  de  $u$  et  $v$  qui réalisent un score d'alignement maximum. On rappelle qu'un mot  $w'$  est *facteur* d'un mot  $w$  si  $w$  peut s'écrire sous la forme  $w = w_1 w' w_2$ .

## 6. $\mathcal{NP}$ -complétude de ORD-TACHES

On s'intéresse au problème suivant qui modélise l'ordonnancement de tâches sur un multiprocesseur : soit  $\Gamma = (S, A)$  un graphe orienté sans circuit, où les sommets représentent les tâches et l'arc  $(x, y)$  signifie que la tâche  $x$  doit être exécutée avant la tâche  $y$ . Une *décomposition compatible* de  $\Gamma$  est une partition de l'ensemble des sommets  $S = S_1 \cup S_2 \cup \dots \cup S_t$  (où  $S_i$  représente l'ensemble des tâches exécutées en parallèle à l'instant  $i$ ), telle que si  $x \in S_i$ ,  $y \in S_j$ , et  $(x, y) \in A$ , alors nécessairement  $i < j$ . La *profondeur* de la décomposition (nombre total d'étapes) est  $t$ , et sa *largeur* (nombre de processeurs nécessaire) est  $l = \max_t \#S_t$ . Étant donné  $\Gamma$ ,  $t$  et  $l$ , le problème ORD-TACHES consiste à décider s'il existe une décomposition de  $\Gamma$  de profondeur au plus  $t$  et de largeur au plus  $l$ . Le but de l'exercice est de montrer que ce problème est  $\mathcal{NP}$ -complet.

On utilisera une réduction de CLIQUE à ORD-TACHES. Soit  $G = (X, E)$  un graphe non orienté à  $n$  sommets et  $m$  arêtes; on construit un graphe orienté  $\Gamma$  dont l'ensemble des sommets est  $S = X \cup E$  et l'ensemble des arcs est  $A = \{(x, e) \in X \times E, x \in e\}$ .

i. On suppose que  $G$  n'a pas de sommet isolé. Soit  $S = S_1 \cup S_2 \cup S_3$  une décomposition compatible de  $\Gamma$  de profondeur 3. Montrer que  $S_1 \cap E = \emptyset$  et que  $S_3 \cap X = \emptyset$ . Soit  $k$  le cardinal de  $S_1 \cap X$ . Montrer que  $\#(S_2 \cap E) \leq k(k-1)/2$ . Que dire des sommets  $S_1 \cap X$  de  $G$  dans les cas où cette inégalité est en fait une égalité ?

ii. On augmente le graphe  $\Gamma$  en un nouveau graphe  $\Gamma'$  avec  $S' = S \cup Y \cup Z \cup T$ ,  $A' = A \cup Y \times Z \cup Z \times T$ , et  $Y, Z$  et  $T$  de cardinaux  $p, q$  et  $r$  respectivement. Combien  $\Gamma'$  a-t-il d'arcs ?

Soit  $k$  tel que  $k \leq n$  et  $k(k-1)/2 \leq m$ . Montrer que l'on peut choisir  $p, q, r, l$  de façon à avoir la propriété suivante :  $G$  admet une clique de taille  $k$  si et seulement si  $\Gamma'$  admet une décomposition compatible en trois sous-ensembles de même taille  $l$ .

iii. En déduire que ORD-TACHES est  $\mathcal{NP}$ -complet.

### 7. $\mathcal{NP}$ -complétude de MAX2SAT

Etant donné un ensemble  $\Phi$  de  $m$  clauses sur les variables booléennes  $x_1, \dots, x_n$ , tel que chaque clause contient au plus deux littéraux, et étant donné un entier  $k$ , le problème MAX2SAT consiste à décider s'il existe une assignation des variables qui satisfasse au moins  $k$  clauses. On se propose de montrer que ce dernier est  $\mathcal{NP}$ -complet.

i. Montrer que MAX2SAT est dans  $\mathcal{NP}$ .

ii. Soit une clause à trois littéraux  $C = x \vee y \vee z$ ; vérifier que dans l'ensemble de 10 clauses donné ci-dessous (faisant intervenir une variable supplémentaire  $t$ ), quelles que soient les valeurs de vérité des variables  $x, y, z$ , au plus 7 des 10 clauses sont satisfaites, et que ce maximum de 7 n'est atteint que si au moins un de  $x, y$  ou  $z$  est vrai :

$$x, y, z, t, \bar{x} \vee \bar{y}, \bar{y} \vee \bar{z}, \bar{z} \vee \bar{x}, x \vee \bar{t}, y \vee \bar{t}, z \vee \bar{t}.$$

ii. En déduire une réduction de 3SAT à MAX2SAT. Conclure.