Cours *conception et analyse d'algorithmes*
Correction du TD 6

1. Recouvrement d'ensembles et de mots

I. Recouvrement par ensembles est $\mathcal{NP}$-complet.

La version de décision du problème RECOUVREMENT PAR ENSEMBLES prend comme données un ensemble fini E, une famille $\mathcal{S}$ de sous-ensembles de E, une function poids $w : \mathcal{S} \mapsto \mathbb{N}$ et un entier W et réponds à la question *existe-t-il un ensemble* $J \subseteq \{1, \ldots, n\}$ *tel que* $\cup_{j\in J} S_j = E$ *et* $\sum_{j\in J} w(S_j) \leq W$?

RECOUVREMENT D'ENSEMBLE est un problème dans $\mathcal{NP}$puisque étant donnée une solution J il est facile de calculer en temps polynomial l'union $\cup_{j\in J} S_j$ en temps polynomial (pour le comparer à E) et la somme $\sum_{j\in J} w(S_j)$ (pour le comparer à W).

On réduit RECOUVREMENT PAR SOMMETS à RECOUVREMENT D'ENSEMBLE comme suit. Supposons donné un graphe $G = (V, E)$ et un entier k. On considère la famille $\mathcal{S} = \{S_v\}_{v\in V}$, où S_v est l'ensemble des arêtes incidente au sommet v. On choisi un poids $w(S_v) = 1$ pour tout sommet v. Avec ces notations, un ensemble $X \subseteq V$ est un ensemble de sommets recouvrant du graphe G si et seulement si $\cup_{v\in X} S_x = E$. De plus, le cardinal de X est égal à son poids $w(X) = \sum_{v\in X} w(S_v)$. Donc RECOUVREMENT PAR SOMMET$(G = (V, E), k)$ donne la même réponse que RECOUVREMENT PAR ENSEMBLE$(E, \mathcal{S}, w, k)$.

Puisque RECOUVREMENT PAR SOMMETS est $\mathcal{NP}$-complet et que la réduction proposée est polynomiale, RECOUVREMENT D'ENSEMBLE est aussi $\mathcal{NP}$-complet.

II Algorithme approché glouton.

1. Soit $J = \{j_1, \ldots, j_p\}$ un recouvrement optimal. On a

$$W_0 = \sum_{j\in J} w(S_j) = \sum_{j\in J} \frac{w(S_j)}{|S_j|} \times |S_j| \geq \min_{j\in J}(\frac{w(S_j)}{|S_j|}) \times \sum_{j\in J} |S_j| \geq \min_{j\in J}(\frac{w(S_j)}{|S_j|}) \times n.$$

Donc $\dfrac{W_0}{n} \geq \min\limits_{j\in J}(\dfrac{w(S_j)}{|S_j|})$.

A l'étape 1, l'algorithme glouton choisit T_1 tel que $c(T_1) = \frac{w(T_1)}{|T_1|}$ soit minimum. Comme ce minimum est inférieur ou égal à $\frac{W_0}{n}$, on obtient bien $\dfrac{w(T_1)}{|T_1|} \leq \dfrac{W_0}{n}$.

2. A l'étape i on peut restreindre le problème à $\overline{F} = E \setminus F$, considérer les sous-ensembles $S'_i = S_i \setminus F$ et supposer que l'on est au début de l'algorithme. L'optimal pour ce problème est $W'_0 \leq W_0$ et par la question précédente on obtient $c(T_i) \leq \dfrac{W'_0}{|\overline{F}|}$ et par suite $c(T_i) \leq \dfrac{W_0}{n - j + 1}$.

3. Le poids de la solution obtenue par l'algorithme glouton est $W = \sum_{i=1}^{k} w(T_i)$. Si l'ensemble T_i contient les éléments x_j pour $j = \alpha_i, \alpha_i + 1, \ldots, \beta_i$, alors la question précédente montre que $c(T_i) = \frac{w(T_i)}{\beta_i - \alpha_i} \leq \frac{W_0}{n-\alpha_i+1}$. Donc $w(T_i) \leq W_0 \frac{\beta_i - \alpha_i}{n-\alpha_i+1} \leq W_0 \sum_{j=\alpha_i}^{\beta_i} \frac{1}{n-j+1}$. Par conséquent $\frac{W}{W_0} \leq \sum_{i=1}^{k} \sum_{j=\alpha_i}^{\beta_i} \frac{1}{n-j+1} = H_n$. L'algorithme est donc H_n approché.

4. On peut prendre comme poids $w(S_i) = 1/i$ pour $i = 1, \ldots, n$ et $w(S_{n+1}) = (1+\varepsilon)$. L'algorithme choisit alors S_n puis S_{n-1} et ainsi de suite jusqu'à S_1 obtenant un poids H_n. La solution optimale est obtenu en prenant S_{n+1} qui a poids $((1+\varepsilon)/n)$. Noter que l'énoncé imposait des poids entiers, il convient donc de multiplier tous les poids par $n!$ pour satisfaire cette condition.

III. Recouvrement de mots.

1. Le mot σ_{12k} est ab^kc et le f optimal est $ab^{k+1}c$.

2. L'algorithme à égalité de longueur de chevauchements pour l'exemple précédent peut choisir σ_{12k} qui donne un ensemble $S = \{ab^kc, b^{k+1}\}$ composé de deux mots qui n'ont qu'un 0-chevauchement d'où la solution obtenue ab^kcb^{k+1}. Cette solution est de longueur $2k+3$ alors que la solution optimale est de longueur $k+2$. Le facteur d'approximation est supérieur ou égal à $\frac{2k+3}{k+2}$ qui aussi proche que l'on veut de 2.

3. Il suffit de prendre pour f la concaténation des mots $\theta_1, \theta_2, \ldots \theta_p$. Comme les S_{θ_i} recouvrent S, tout f_j est élément d'un des S_{θ_i}, donc facteur du θ_i correspondant. Le poids W_0 de la solution optimale au problème Q est égal à la somme des longueurs des mots θ_i, qui est exactement la longueur de f.

4. Soit $k \in \{1, \ldots, n\}$. On va montrer que f_k est dans un des sous-ensemble S_{π_i}. Si k est égal à l'un des b_i ou des e_i alors c'est clairement le cas, sinon puisque $1 = b_1 < e_1 < b_2 < e_2 < \cdots < b_t < e_t = n$ et que $b_i = e_{i-1} + 1$, il existe un indice i tel que $b_i < k < e_i$. Donc, le facteur f_k commence après f_{b_i} et avant f_{e_i}. Comme on suppose que les f_i ne sont pas facteurs les uns des autres, le facteur f_k termine avant f_{b_i}. Donc f_k est contenu dans le sous mot $\pi_i = \sigma_{b_i e_i k_i}$ et il appartient à l'ensemble S_{π_i}. Les S_{π_i} constituent donc un recouvrement de S.

5. Soit $i \in \{1, \ldots, t-2\}$. Par définition, le facteur $f_{b_{i+1}}$ commence après f_{e_i} dans le mot f (car $b_{i+1} > e_i$) donc $f_{b_{i+1}}$ termine après f_{e_i} (car $f_{b_{i+1}}$ n'est pas facteur de f_{e_i}). D'autre part, $f_{b_{i+1}}$ termine avant que $f_{b_{i+2}}$ ne commence. Donc f_{e_i} termine avant que $f_{b_{i+2}}$ ne commence. Donc π_i et π_{i+2} ne s'intersectent pas.

On en déduit que $\sum_{i \text{ pair}} l(\pi_i) \leq l(f)$ et $\sum_{i \text{ impair}} l(\pi_i) \leq l(f)$. Donc $\sum_i l(\pi) \leq 2l(f)$.

6. On considère l'algorithme qui remplace le problème P sur les mots par le problème Q correspondant sur les ensembles, puis qui résout Q par l'algorithme glouton de la partie II, enfin qui concatène les mots u correspondants aux S_u de la solution trouvée.

D'après les questions II.3,II.4,II.5, on sait que s'il existe une solution de poids l pour le problème P de recouvrement de mots, alors il existe aussi une solution de poids au plus $2l$ obtenu en passant le problème Q correspondant. On obtient une solution de poids au plus $2lH_n$ en utilisant un algorithme glouton pour résoudre le problème P. Notre algorithme est donc $2H_n$ approché.

2. Réseau avec relais, NP-complétude, approximation

A. Un algorithme d'approximation pour Réseau avec relais.

1. Étant donné un arbre connectant optimal T, son parcours (par exemple en profondeur) permet de construire un cycle hamiltonien dans G_S. Le poids d'une arête (x, y) de ce cycle est majoré par la longueur du chemin de x à y dans le parcours de T. Ce parcours utilise au total exactement 2 fois chaque arêtes de T donc le poids du cycle hamiltonien construit dans G_S est au plus $2|T|$ et, en enlevant une arête, c'est aussi une borne sur le poids d'un arbre couvrant de poids minimum dans G_S.
2. Un arbre couvrant T_S de poids minimum de G_S permet de construire un ensemble d'arêtes F connectant les sommets de S à un coût au plus $p(T_S)$: on prend l'union des chemins minimaux de G associés aux arêtes de T_S, et on élimine encore des arêtes s'il y a des cycles.

   La complexité de l'algorithme dépend du cardinal de S : il faut calculer les distances deux à deux entre ces sommets ($|S|^2 \cdot (|E| + |X| \log |X|)$ ou $|X|^3$) puis appliquer un algorithme de calcul d'un arbre couvrant de poids minimum ($O(|S|^2 \log |S|)$).

   Le réseau de moindre coût avec relais s'appelle aussi arbre de Steiner.

B. NP-complétude du problème Réseau avec relais.

1. On réalise un parcours en profondeur du graphe obtenu qui permet de s'assurer que tous les sommets de S sont connectés entre eux.
2. Étant donné un ensemble Y avec $|Y| = 3n$ et une collection C de triplets de Y on construit une instance de Réseau avec relais en prenant comme ensemble de sommets $X = \{x_0\} \cup C \cup Y$, comme ensemble d'arêtes $E = \{(x_0, c) \mid c \in C\} \cup \{(c, y) \mid c \in C,\, y \in c\}$, comme ensemble de sommets à relier $S = \{x_0\} \cup Y$ et comme taille de réseau à ne pas dépasser $k = 4n$.

   Un certificat de réseau avec relais sur ce graphe à $4n$ arêtes contient exactement une arête (c, y) pour tout $y \in Y$ (ce qui donne $3n$ arêtes) et n arêtes entre x_0 et les c correspondants. L'ensemble des c forme une partition de $|Y|$ de taille n.