Master Informatique Fondamentale (1ère année)
Algorithmes d'approximation — **TD3**

Dominer, couvrir, colorier...

22 octobre 2004

Exercice 1 : Dominant. On appelle ensemble *dominant* d'un graphe $G = (V, E)$ un ensemble de sommets $D \subseteq V$ tel que $\forall v \in V \setminus D$, il existe $d \in D$ avec $vd \in E$.

Question 1. *Le problème de trouver un dominant est-il le même que celui de trouver une couverture par sommets ? Sinon, en quoi sont-ils différents ?*

Réponse : Si le graphe possède des sommets isolés, on ne peut pas trouver de dominant. Soit $G = (V, E)$ un graphe non orienté sans sommet isolés. Soit C une couverture par sommets de taille k. Montrons qu'on peut en déduire facilement un dominant de même taille k. Par définition, on a $C \subseteq V$ tel que $\forall e \in E, \exists (u, v) \in C \times V$, tel que $e = (u, v)$. Soit $v\ V \setminus C$. Comme le graphe est sans sommet isolé, il existe $e \in E$ tel que $e = (v, d)$, avec $d \in V$. Comme C est une couverture par sommet, on a $d \in C$. Ainsi, C est bien un dominant de taille k.
En revanche, un dominant n'est en général pas une couverture par sommets : dans un triangle, il faut au moins 2 sommets pour une couverture par sommets, alors qu'un seul suffit pour résoudre un dominant. Ce sont tous les deux des problèmes NP-difficiles. □

Exercice 2 : Acheminement de biscuits. Considérons un graphe $G = (V, E)$ muni d'une fonction de coût positive sur les arêtes et dans lequel on distingue deux sous-ensembles (disjoints) de V : $\mathcal{E}$ l'ensemble des entrepôts de biscuits et $\mathcal{D}$ l'ensemble des destinataires. Le problème est de trouver un sous-graphe de G de coût minimum dans lequel tout destinataire est relié à l'un (au moins) des entrepôts.

Question 2. *Montrer que ce problème, réduit aux instances telles que $\mathcal{E} \cup \mathcal{D} = V$, est polynomial.*

Réponse :

- On peut déjà remarquer que le graphe cherché est acyclique : s'il existe un cycle, supprimer n'importe quel arête du cycle ne change pas la connexité et réduit le coût du graphe.
- Il n'est pas réducteur de supposer le graphe G connexe.
- 1^{er} cas : $\mathcal{E} = \{s_0\}$
  D'après la remarque précédente, on cherche un arbre T qui passe par tous les destinataires et par au moins un entrepôt (exactement un car il n'y en a qu'un). Comme on a $\mathcal{E} \cup \mathcal{D} = V$, on cherche donc un arbre couvrant tous les sommets et de poids minimum (Minimum Spanning Tree, MST). Or on sait calculer MST en temps polynomial : on trie les arêtes par poids croissants $\{e_1, ..., e_m\}$, on les consifère les unes après les autres et on prend celles qui ne créent pas de cycle (algorithme de Kruskal).
- 2^{me} cas : $\mathcal{E} = \{s_0, ..., s_p\}$ On va généraliser la méthode précédente. Notons E' l'ensemble obtenu en rajoutant aux arêtes d'origine E des arêtes de poids nul entre les entrepôts : $E' = E \bigcup \{(s_i s_j); i, j \leq p, i \neq j\}$, et on pose $w(e) = 0$ pour $e \in E' \setminus E$. L'ensemble des entrepôts ainsi reliés forme un "meta-entrepôt" $\mathcal{ME}$. Soit T un MST couvrant $G' = (V, E')$. Montrons que T est bien la solution à notre problème.
  - Tous les sommets de $\mathcal{D}$ sont reliés à $\mathcal{E}$ par les arêtes de T (par définition de T).

– Soit S une solution du problème, tel que $w(S) < w(T)$ On sait que S est une forêt d'arbres, telle que chaque arbre $T' \in S$ possède au moins un sommet de $\mathcal{E}$. Relions entre eux les arbres de S par les arêtes de $E' \setminus E$, sans créer de cycle. On obtient un arbre couvrant G, de poids strictement inférieur à T => Contradiction avec la minimalité de T. T est donc bien une solution optimale.

□

Question 3. *Montrer que le problème général est NP-difficile et proposer une* 2*-approximation.*

Réponse : On va utiliser le problèmes des arbres de Steiner (cf. le TD1) pour effectuer la réduction.
Rappel du problème des arbres de Steiner :
Etant donné un graphe G non orienté, muni d'une fonction de coût positive sur les arêtes, et donc certains sommets sont étiquetés $Requis$, trouver un arbre de coût minimum dans G contenant tous les sommets étiquetés $Requis$.
On pose $Requis = \{\mathcal{ME}, \mathcal{D}\}$. On remarque qu'il n'y a pas besoin de changer les poids des arêtes. Une solution au problème de Steiner sera également une solution au problème de la biscuiterie, et réciproquement.

– Une 2-approximation d'une solution optimale au problème de Steiner sera bien une forêt contenant $\mathcal{ME}$ et tous les destinataires.
– Une 2-approximation du problème de la biscuiterie doit également passer par tous les destinataires et au moins un entrepôt (donc par $\mathcal{ME}$) et est donc une 2-approximation du problème des arbres de Steiner.

L'utilisation de l'algorithme d'approximation vu au TD1 nous donne une 2-approximation pour le problème de Steiner, et donc une 2-approximation pour le problème de la biscuiterie.
Passons à la réduction, pour montrer que la biscuiterie est bien un problème NP-difficile.

– On peut transformer toute instance des arbres de Steiner en une instance de la biscuiterie : on prend un sommet s de $\mathcal{R}$; on pose $\mathcal{E} = \mathcal{ME} = \{s\}$ et $\mathcal{D} = \mathcal{R} \setminus \mathcal{E}$. On peut garder la même fonction de poids dans les problèmes.
– Une solution du problème de Steiner va passer par tous les sommets requis et par $\mathcal{E}$ donc par au moins un entrepôt et sera minimal, cette solution est donc également une solution pour la biscuiterie. La minimalité est conservée, car la fonction de poids est conservée, et les entrepôts sont reliés par des arêtes de poids nul. De la même façon, on montre que toute solution de la biscuiterie est également solution des arbres de Steiner.

□

Exercice 3 : Coloriage de sommets. Étant donné un graphe $G = (V, E)$, on cherche à colorier ses sommets avec un nombre minimum de couleurs et en respectant la contrainte que 2 sommets adjacents ne sont jamais de la même couleur.

Question 4. *Donner un algorithme glouton qui colorie* G *avec* $\Delta + 1$ *couleurs, où* Δ *est le degré maximum des sommets de* G*.*

Réponse : L'algorithme glouton à utiliser est simple, il suffit de colorier les sommets un par un (triés par un ordre quelconque) et à chaque fois on attribue au sommet la première couleur libre. Comme chaque sommet a au plus Δ voisins, au pire Δ couleurs sont déjà utilisées par ses voisins et la $(\Delta + 1)$ème est libre. □

Question 5. *Montrer que dans un graphe* 3*-coloriable le sous-graphe induit par les voisins d'un sommet donné est toujours bi-parti.*

Réponse : Le sous-graphe induit par les voisins d'un sommet donné est 2-coloriable, car il n'y a que 2 couleurs (au plus) parmi les voisins de tout sommet. Or tout graphe est 2-coloriable si et seulement s'il est biparti, donc on a bien le résultat cherché. □

Question 6. *Donner un algorithme en temps polynomial pour colorier avec $O(\sqrt{n})$ couleurs tout graphe 3-coloriable. (Indication : On pourra séparer les sommets en deux ensembles : ceux de degré supérieur à une certaine fonction $f(n)$ et ceux de degré inférieur).*

Réponse : On va séparer les sommets de $G = (V, E)$ en deux ensembles disjoints, ceux de degré supérieur à $\sqrt{n}$ et ceux de degré strictement inférieur à $\sqrt{n}$. Notons $V_1 = \{v \in V | deg(v) \geq \sqrt{n}\}$ et $V_2 = \{v \in V | deg(v) < \sqrt{n}\}$.

- 1^{ere} étape : Tant que $V_1 \neq \emptyset$ faire : On prend un élément v de V_1, on le colorie ainsi que tous ses voisins non déjà coloriés avec 3 couleurs (c'est possible car le graphe est 3-coloriable). On supprime tous les sommets ainsi coloriés du graphe et on met à jour V_1 et V_2. Cette boucle est faite au plus $\sqrt{n}$ fois, car on supprime à chaque tour $\sqrt{n}$ sommets (et il y en a n au total). On utilise donc au plus $3 * \sqrt{n}$ couleurs. Cette étape est polynomiale.
- 2^{nde} étape : Le graphe restant après la 1^{ere} étape est donc un graphe $G = (V_2, E')$ dont chaque sommet a au plus $\Delta = \sqrt{n}$ voisins. D'après la question 4, il est possible de le colorier avec $\sqrt{n} + 1$ couleurs, de façon polynomiale.

Finalement, on bien colorié G avec $4 * \sqrt{n} + 1 = O(\sqrt{n})$ couleurs, et l'algorithme proposé est polynomial. □

Exercice 4 : Couverture maximum. Étant donné un univers U à n éléments muni d'une fonction de poids p positive, une famille $\mathcal{S} = \{S_1, \ldots, S_p\}$ de sous-ensembles de U et un entier k, le problème est de sélectionner k ensembles de $\mathcal{S}$ en maximisant le poids total des éléments couverts. On étudie l'algorithme glouton qui consiste à sélectionner à chaque étape l'ensemble le plus "rentable" jusqu'à ce que k ensembles aient été sélectionnés.

Question 7. *Écrire cet algorithme et vérifier qu'il est polynomial.*

Réponse : L'ensemble le plus "rentable" est *a priori* celui qui maximise la somme des poids de ses éléments non déjà couverts.

$\mathcal{S} \leftarrow \{S_1, \ldots, S_p\}$	$O(p)$
$\mathcal{R} \leftarrow \emptyset$ (* la réponse *)	$O(1)$
pour i de 1 à p : $S'_i \leftarrow S_i$; $p(S'_i) \leftarrow p(S_i)$	$O(np)$
pour j de 1 à k :	$\times k$
trouver S'_j de poids maximum dans $\mathcal{S}$	$O(p)$
$\mathcal{R} \leftarrow \mathcal{R} \cup S'_j$	$O(1)$
pour tout $x \in S'_j$:	$\times O(n)$
pour tout $S'_i \in \mathcal{S}$:	$\times O(p)$
$S'_i \leftarrow S'_i \setminus \{x\}$	$O(n)$
$p(S'_i) \leftarrow p(S'_i) - p(x)$	$O(1)$
$\mathcal{S} \leftarrow \mathcal{S} \setminus \{S'_j\}$	$O(p)$

Cet algorithme renvoie une partie de $\mathcal{S}$ à k éléments.
Il fait bien ce qu'on veut car à chaque étape, chaque S'_i vaut S_i privé de ses éléments déjà couverts.
Sa complexité en temps est en $O(kpn^2)$ (polynomiale). □

On note $\left(S_{g_i}\right)_{1 \leq i \leq k}$ la suite d'ensembles sélectionnés par l'algorithme glouton.

Question 8. *Montrer que pour tout $1 \leq l \leq k$:*

$$p\left(\bigcup_{i=1}^{l} S_{g_i}\right) - p\left(\bigcup_{i=1}^{l-1} S_{g_i}\right) \geq \frac{p(OPT) - p\left(\bigcup_{i=1}^{l-1} S_{g_i}\right)}{k}.$$

Réponse :

On se place à l'étape l ; en particulier S'_i est pris dans la boucle quand $j = l$.
On note $\Sigma_t = \bigcup_{i=1}^{t} S_{g_i}$. On a ainsi d'emblée : $p(\Sigma_l) - p(\Sigma_{l-1}) = p(S'_{g_l})$, et il s'agit de montrer que

$$p(OPT) \leq p(\Sigma_{l-1}) + kp(S'_{g_l}).$$

OPT s'écrit sous la forme $\bigcup_{i=1}^{k} S_{h_i}$.

$$OPT \subseteq \bigcup_{i=1}^{k} S_{h_i} \cup \Sigma_{l-1} = \Sigma_{l-1} \cup \bigcup_{i=1}^{k} (S_{h_i} \setminus \Sigma_{l-1})$$

$$\text{donc} \quad p(OPT) \leq p(\Sigma_{l-1}) + \sum_{i=1}^{k} p(S_{h_i} \setminus \Sigma_{l-1}) = p(\Sigma_{l-1}) + \sum_{i=1}^{k} p(S'_{h_i})$$

Par choix glouton de S_{g_l}, on a $\forall i,\ p(S'_{h_i}) \leq p(S'_{g_l})$, donc $p(OPT) \leq p(\Sigma_{l-1}) + kp(S'_{g_l})$. □

Question 9. *Montrer que pour tout* $1 \leq l \leq k$ *:*

$$p\left(\bigcup_{i=1}^{l} S_{g_i}\right) \geq \left(1 - \left(1 - \frac{1}{k}\right)^{l}\right) p(OPT).$$

En déduire une borne sur le facteur d'approximation de l'algorithme.

Réponse :

De $p(\Sigma_l) - p(\Sigma_{l-1}) \geq \frac{1}{k}(p(OPT) - p(\Sigma_{l-1}))$ on déduit que $p(\Sigma_l) \geq \frac{p(OPT)}{k} + (1 - \frac{1}{k})p(\Sigma_{l-1})$. Posons $u_l = \frac{k}{p(OPT)} p(\Sigma_l)$. On a alors

$$\forall 1 \leq l \leq k,\ u_l \geq 1 + \alpha u_{l-1} \quad \text{avec} \quad \alpha = 1 - \frac{1}{k}$$

Par récurrence immédiate,

$$\forall 1 \leq j \leq l,\ u_l \geq \sum_{i=0}^{j-1} \alpha^i + \alpha^j u_{l-j}$$

En particulier pour $j = l$: $u_l \geq \sum_{i=0}^{l-1} \alpha^i + \alpha^l u_0$.
On revient maintenant au problème initial : $u_0 = 0$ et $u_l = \frac{k}{p(OPT)} p(\Sigma_l)$, donc

$$p(\Sigma_l) \geq \frac{p(OPT)}{k} \sum_{i=0}^{l-1} \left(1 - \frac{1}{k}\right)^{i} = p(OPT) \left(1 - \left(1 - \frac{1}{k}\right)^{l}\right)$$

La réponse de l'algorithme est Σ_k ;

$$p(\Sigma_k) \geq p(OPT)(1 - (1 - \frac{1}{k})^k)$$

Avec l'identité $\ln(1 + x) \leq x$ appliquée à $x = (1 - \frac{1}{k})$, on a $\ln((1 - \frac{1}{k})^k) = k \ln(1 - \frac{1}{k}) \leq -1$, donc $(1 - \frac{1}{k})^k \leq e^{-1}$ et

$$p(\Sigma_k) \geq p(OPT) \left(1 - \frac{1}{e}\right)$$

L'algorithme est donc une $\left(1 - \frac{1}{e}\right)$ - approximation.
Remarque : le facteur d'approximation est inférieur à 1, mais c'est normal étant donné qu'on a un problème de maximisation. □

♣