# Corrigé de la PC7

7 mars 2007

**0. Programmation linéaire, algorithmes probabilistes**

On considère le problème SET-COVER pour lequel la donnée est un ensemble $E = \{1, 2, \ldots, n\}$ et une famille $\mathcal{F}$ de sous-ensembles $F_1, F_2, \ldots, F_m$ de $E$ dont l'union est égale à $E$. On suppose de plus que chaque élément $i$ est contenu dans exactement $k$ sous-ensembles $F_i$. Il s'agit de trouver un nombre minimum de sous-ensembles $F_i, i \in I$ tels que $\bigcup_{i \in I} F_i = E$.

**1.** Pour exprimer le problème sous forme d'un programme linéaire en nombres entiers on exprime que chaque élément $i$ doit être dans l'un des $F_j$ soit pour chaque $i$ soit $F_{j_1}, F_{j_2}, \ldots, F_{j_k}$ les sous-ensembles qui contiennent $i$ on doit alors avoir $\sum_{i=1}^{k} x_{j_i} \geq 1$ et il faut rendre minimum $\sum_{i=1}^{m} x_i$

Pour l'exemple suivant $E = \{1, 2, 3, 4, 5\}, F_1 = \{1, 3, 5\},\ F_2 = \{2, 3\},\ F_3 = \{1, 2, 4\},\ F_4 = \{4, 5\},\ F_5 = \{1, 2, 4\},\ F_6 = \{3, 5\}$.

on alors :

$$x_1 + x_3 + x_5 \geq 1, \quad x_2 + x_3 + x_5 \geq 1, \quad x_1 + x_2 + x_6 \geq 1, \quad x_3 + x_4 + x_5 \geq 1 \quad x_1 + x_4 + x_6 \geq 1$$

et il faut minimiser

$$x_1 + x_2 + x_3 + x_4 + x_5$$

**2.** Soit $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m$ la solution au problème lorsqu'on relache la condition nombres entiers pour autoriser des rationnels. L'optimum du cas entier est supérieur ou égal à celui du cas rationnel puisqu'il y a une condition supplémentaire, ainsi l'optimum entier est supérieur ou égal à

$$\hat{\sigma} = \hat{x}_1 + \hat{x}_2 + \ldots + \hat{x}_m$$

et comme l'optimum est entier il est aussi supérieur ou égal à la partie entière par excès de $\hat{\sigma}$.

**3.** On applique la méthode suivante pour résoudre notre problème à partir de la solution en nombres rationnels $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m$ : déterminer si $i \in I$ avec probabilité $\hat{x}_i$. Cette méthode est de type Monte-Carlo car on n'obtient pas toujours une solution au problème.

La probabilité pour que $F_{j_1}$ ne soit pas choisi est $1-\hat{x}_{j_1}$ ainsi la probabilité pour qu'un élément $j$ donné appartenant à $F_{j_1}, F_{j_2}, \ldots, F_{j_k}$ n'appartienne à aucun des $F_i$ pour $i \in I$ :

$$\Pi_{i=1}^{k}(1-\hat{x}_{j_i})$$

**4.** La probabilité que $j$ soit dans un des $F_i, i \in I$ est $1-\Pi_{i=1}^{k}(1-\hat{x}_{j_i})$. Puisque la somme des $\hat{x}_{j_i}$) est supérieure ou égale à 1 le produit des $(1-\hat{x}_{j_i}$ est majoré par le cas où tous les nombres sont égaux soit :

$$(1-\frac{1}{k})^k \leq \frac{1}{e}$$

La probabilité qu'un $j$ donné soit dans l'un des $F_{j_i}$ est donc supérieure ou égale à $(1-\frac{1}{e})$. Par ailleurs la moyenne des valeurs obtenues pour la somme $\sum_{i=1}^{m} x_i$ est l'optimum réel $\hat{\sigma}$.

**5.** L'idée est ici de répeter $log(n)$ fois le tirage aléatoire permettant de décider avec probabilité $\hat{x}_{j_i}$ de retenir $F_{i_j}$ dans la solution et de le retenir si le tirage est favorable au moins une fois. La solution obtenue est alors en moyenne $log(n)$ fois l'optimum réel et la probabilité que tous les éléments de $E$ soient couverts est supérieure à $(1-\frac{1}{e})$ donc à $\frac{1}{2}$.

### 1. Algorithme probabiliste pour 2-SAT

i) À chaque étape, il faut déterminer une clause non satisfaite ; cela impose en particulier dans le cas le pire d'évaluer toutes les clauses, soit $O(p)$ opérations. Comme l'algorithme comporte $m$ étapes, cela entraîne que la complexité de l'algorithme est $O(mp)$. Il est bien clair que l'algorithme peut se tromper ; prenons le cas des deux clauses $\overline{x} \vee y$ et $x \vee z$. Si l'on part de l'affectation $(x, y, z) = (0, 0, 0)$ et que l'on suppose qu'à chaque étape $x$ est modifié, les deux clauses sont alternativement l'une vraie et l'autre fausse ; pourtant l'affectation $(0, 1, 1)$ (par exemple) prouve que l'ensemble de clauses est satisfiable.

ii) Si $v_0 = v$, toutes les clauses sont satisfaites et l'algorithme termine, donc $E(0) = 0$. Si $v_0 = \overline{v}$, l'algorithme va modifier une quelconque des variables ; l'affectation $v'$ obtenue aura alors $n-1$ variables ayant des valeurs différentes de leurs valeurs dans $v_0$, et le nombre moyen de modifications restant à faire sera donc $E(n-1)$, d'où $E(n) = 1 + E(n-1)$.

Supposons donnée une assignation $x$ telle que $d(x, u) \leq i$. Soit $c$ la clause non satisfaite choisie. Une des deux variables de $c$ au moins a une valeur différente de celle qu'elle a dans l'affectation $u$ ; par suite, avec probabilité au moins $1/2$, le changement de $x$ en $x'$ entraîne $d(x', u) = d(x, u) - 1$, et avec probabilité au plus $1/2$, $d(x', u) = d(x, u) + 1$. Par suite, le nombre d'itérations moyen partant de $x$ est au plus $1 + (E(i-1) + E(i+1))/2$, et comme cette borne ne dépend pas de $x$, on a bien $E(i) \leq 1 + (E(i-1) + E(i+1))/2$.

iii) La dernière identité se réécrit en $E(i) - E(i-1) \leq E(i+1) - E(i) + 2$, d'où $E(i+1) - E(i) \leq 2(n-i) + E(n) - E(n-1) = 2(n-i) + 1$, et $E(i) \leq i + \sum_{k=0}^{i-1} 2(n-i) \leq i(2n-i)$, soit $E(n) \leq n^2$.

iv) L'inégalité de Markov entraîne que la probabilité que le nombre d'affectations nécessaire soit plus grand que $2E(n) = 2n^2$ est au plus $1/2$.

v) On a toujours $E(0) = 0$, ainsi que $E(n) = 1 + E(n-1)$, pour les mêmes raisons que précédemment. En revanche, à chaque fois que l'on modifie une des variables dans une clause invalide, on a “deux chances sur trois” de modifier une variable qui avait une valeur correcte, contre “une chance sur trois” de modifier une variable incorrecte). Cela se traduit par $E(k) = 1+(E(k-1)+2E(k+1))/3$. Par suite, $2E(k+1)-3E(k)+E(k-1)+3 = 0$, et donc $E(k) - E(k-1) = 2(E(k+1) - E(k)) + 3$. Posant $a_i = E(n-i) - E(n-i-1)$, on a bien $a_0 = 1$ et $a_{i+1} = 2a_i + 3$, soit encore $a_i = 2^{i+2} - 3$. Par suite $E(n) = E(n-k) + \sum_{i=0}^{k-1}(2^{i+2}-3) = E(n-k)+(2^{k+2}-4)-3k$. En faisant $k = 0$, il vient $E(n) = 2^{n+2}-4-3n$. Par suite, 3-SAT reste exponentiel même par cette approche : si l’on veut se donner une probabilité $1/2$ de trouver la bonne solution, l’inégalité de Markov recommande d’utiliser $m \approx 2^{n+3}\ldots$

**2. Coupure minimale**

i) Soit $C_0$ une coupure du graphe contracté. Montrons que la coupure $C_0$ est aussi une coupure dans le graphe initial. Il est bien clair, par récurrence, que l’on peut se restreindre au cas où l’on n’effectue qu’une contraction.

En effet, la suppression de $C_0$ rend $G'$ non connexe, $G'$ étant le graphe dans lequel les sommets $u$ et $v$ sont fusionnés. Soient $C_1$ et $C_2$ deux composantes connexes distinctes de $G'-C_0$, avec $u$–$v \in C_1$. La suppression de $C_0$ dans $G$ conduit alors au même graphe, excepté que l’unique sommet $u$–$v$ est remplacé par deux sommets reliés par plusieurs arêtes. Si l’on suppose que le graphe $G - C_0$ est connexe, il existe une arête allant de $u$ ou de $v$ vers $C_2$ ; mais dans le graphe contracté, cette arête devient une arête de $u$–$v$ vers $C_2$, impossible.

ii) Il suffit de montrer que tous les sommets de $G$ ont un degré au moins égal à $k$. Dans le cas contraire, il existe un sommet de degré strictement inférieur à $k$ et la suppression des arêtes sortantes conduit à un graphe non connexe ; il existe donc dans ce cas une coupe de taille strictement inférieure à $k$, contrairement à l’hypothèse. De cette façon, il s’ensuit que $G$ a au moins $kn/2$ arêtes (chaque arête est comptée deux fois).

iii) Soit $C$ une coupe minimale. Remarquons tout d’abord que si au cours de l’algorithme une arête ayant les mêmes extrémités qu’une arête de $C$ est contractée, alors soit c’est une arête de $C$, soit une arête de $C$ a déjà été contractée ; en effet, si $(u, v)$ est cette arête, alors $u$ et $v$ sont dans deux composantes connexes distinctes de $G - C$. Donc soit à l’étape courante, $u$ et $v$ sont toujours dans deux composantes connexes distinctes de $G - C$, et toute arête entre $u$ et $v$ est une arête de $C$ (qui est minimale), soit $u$ et $v$ sont dans la même composante connexe de $G - C$, ce qui signifie qu’une arête de $C$ a déjà été contractée.

Cela implique que tant que l’on n’enlève pas d’arête de la coupe minimale, cette dernière reste de taille égale à $k$.

À la première étape, la probabilité que l’arête enlevée soit dans $C$ est au plus $k/(kn/2) \leq 2/n$. La probabilité que l’arête enlevée ne soit pas dans $C$ est donc au moins $1 - 2/n$.

Lors de la deuxième étape, le graphe a alors une coupe minimale de taille $k$, donc au moins $k(n-1)/2$ arêtes. Il s’ensuit que la probabilité de ne pas enlever d’arête de $C$ est au moins $1 - k/(k(n-1)/2) = 1 - 2/(n-1)$. On voit facilement par récurrence que la probabilité à la $i$-ème étape est de $2/(n-i+1)$, et donc

$$P(\text{pas d'arête de } C \text{ contractée avant la } l\text{-ième étape}) \geq \prod_{i=1}^{n-2}\left(1 - \frac{2}{n-i+1}\right) = \frac{2}{n(n-1)}.$$

Noter que la preuve rigoureuse passe par l'utilisation de probabilités conditionnelles.

iv) La complexité de l'étape de base de l'algorithme est de $O(d)$, où $d$ est le degré maximal des sommets du graphe, ou en terme du nombre d'arêtes en $O(n^2)$.

Notons que si l'algorithme est itéré $f(n)$ fois, et que l'on conserve le plus petit des résultats, la probabilité que ce résultat ne soit pas bon est au plus de $(1-2/n(n-1))^{f(n)}$, soit au plus de $\exp(\log(1-2/n(n-1))f(n))$. En utilisant l'inégalité $\log(1-u) \leq -u$, on voit que la probabilité est d'échec est au plus de $\exp(-2f(n)/n(n-1))$. En particulier, en prenant $f(n) = n(n-1)\log n$, on trouve que cette probabilité est au plus de $1/n^2$, pour une complexité $O(n^4 \log n)$.

**3) Vérification de calculs matriciels**

i) Le calcul d'une coordonnée de $Ax$ se fait via la formule $\sum_j A_{ij}x_j$, et donne donc lieu à $O(n)$ opérations, soit $O(n^2)$ opérations. Il s'ensuit que l'on peut se permettre de faire $O(n^{0.376})$ calculs de $Ax$ sans dépasser le temps de calcul du produit par la méthode rapide. Cela exclut en particulier de vérifier l'exactitude sur les éléments d'une base, ce qui garantirait l'exactitude (mais la complexité redevient $O(n^3)$).

ii) Il est bon de préciser ce que l'on entend par probabilité : $r$ est choisi au hasard dans $\{0,1\}^n$, et c'est cet espace que l'on probabilise. L'égalité $ABr = Cr$ équivaut à $(AB-C)r = 0$, soit encore $r \in Ker(AB-C)$. Montrons que tout sous-ensemble de $\{0,1\}^n$ de taille au moins $2^{n-1}+1$ n'est incluse dans aucun sous-espace vectoriel strict. On peut supposer sans perte de généralité que ce dernier est un hyperplan, et est donc défini par une forme linéaire $\sum \alpha_i x_i$. Fixons un $i$, et considérons la projection de notre ensemble sur $\{0,1\}^{n-1}$ obtenue en oubliant la $i$-ème composante. Comme $\{0,1\}^{n-1}$ ne contient que $2^{n-1}$ éléments, il s'ensuit que deux éléments du sous-ensemble de départ ont toutes leurs coordonnées égales sauf la $i$-ème ; en soustrayant la forme linéaire en ces deux points, on voit que $\alpha_i = 0$. Il s'ensuit que la forme linéaire est nulle. Par suite, si $AB - C \neq 0$, le nombre de $r$ de $\{0,1\}^n$ qui annulent $AB - C$ est au plus $2^{n-1}$, ce qui revient à dire que pour un $r$ choisi au hasard, la probabilité est au plus $1/2$.

iii) Après $l$ essais, il est facile de voir que la probabilité d'échec est au plus $1/2^l$, et donc pour $l = \log_2 n$, au plus $1/n$.