Cours *conception et analyse d'algorithmes*
Correction du TD 7

1. Programmation linéaire, algorithmes probabilistes

i. 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 $\sum_{i=1}^{m} x_i$.

Sur l'exemple proposé, il faut minimiser $x_1 + x_2 + x_3 + x_4 + x_5$ sous les contraintes

$$x_1 + x_3 + x_5 \geq 1, \quad x_2 + x_3 + x_5 \geq 1, \quad x_1 + x_2 + x_6 \geq 1, \quad x_3 + x_4 + x_5 \geq 1 \text{ et } x_1 + x_4 + x_6 \geq 1.$$

ii. Soit $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m$ la solution au problème lorsqu'on relâche 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 $\hat{\sigma}$.

iii. Cette méthode est de type Monte-Carlo car on n'obtient pas toujours une solution au problème.

On note X_i l'évènement *le sous-ensemble F_i est sélectionné* et Y_j l'évènement *j appartient à l'un des sous-ensembles F_j*.

On a $Y_j = \cup_i$ tel que $_{j \in F_i} X_i$ donc

$$P(Y_j) = 1 - P(\bigcap_{i \text{ tel que } j \in F_i} \overline{X_i}).$$

Par l'indépendance des évènements F_i on en déduit

$$P(Y_j) = 1 - \prod_{i \text{ tel que } j \in F_i} (1 - \hat{x}_i).$$

iv. Par une inégalité de concavité du logarithme on obtient que

$$\prod_{i \text{ tel que } j \in F_i} (1 - \hat{x}_i) \leq \left(\frac{1}{k} \sum_{i \text{ tel que} j \in F_i} 1 - \hat{x}_i \right)^k.$$

Comme $\sum_{i \text{ tel que } j \in F_i} \hat{x}_i \geq 1$ on a

$$\prod_{i \text{ tel que } j \in F_i} (1 - \hat{x}_i) \leq (\frac{k-1}{k})^k$$

et par suite

$$P(Y_j) = 1 - \prod_{i \text{ tel que } j \in F_i} (1 - \hat{x}_i) \geq 1 - (\frac{k-1}{k})^k = 1 - \exp(k \log(1 - 1/k)) \geq 1 - \frac{1}{e}.$$

v. L'idée est ici de répéter $f = O(\log(n))$ fois le tirage aléatoire permettant de décider avec probabilité $\hat{x}_i$ et de retenir le sous-ensemble F_i dans la solution si le tirage est favorable au moins une fois. Ainsi la probabilité de l'évènement X_i devient $1-(1-\hat{x}_i)^f$ et la probabilité de Y_j est au moins $1 - e^{-f}$.

On a pas indépendance des évènements Y_j donc pour minorer la probabilité que tous les Y_j soit vrai nous allons utiliser l'inégalité de Markov. On calcule :

$$E(\sum_{j=1}^{n} 1 - \mathbb{1}_{Y_j}) = \sum_{j=1}^{n} E(1 - \mathbb{1}_{Y_j}) \leq ne^{-f},$$

où $\mathbb{1}_{X_i}$ est la fonction indicatrice de X_i qui vaut 1 si X_i est vrai et 0 sinon. Par l'inégalité de Markov, on obtient

$$1 - P(\bigcap_{j=1}^{n} Y_j) = P(\sum_{j=1}^{n} 1 - \mathbb{1}_{Y_j} \geq 1) \leq E(\sum_{j=1}^{n} 1 - \mathbb{1}_{Y_j}) \leq ne^{-f},$$

où $\mathbb{1}_{Y_j}$ désigne l'indicatrice de l'évènement Y_j. On en déduit que $P(\bigcap_{j=1}^{n} Y_j)$ est supérieur a $1/2$ dès que $f \geq \log(n) + \log(2)$.

Par ailleurs, l'espérance du nombre de sous-ensembles sélectionnés est $E(\sum_i X_i) \leq f \sum_i \hat{x}_i$, soit $O(\log(n))$ fois l'optimum.

2. 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 au plus $E(n - 1)$, 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, $E(i)$ étant par définition une fonction croissante de i, 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 au plus "deux chances sur trois" de modifier une variable qui avait une valeur correcte, contre seulement au moins "une chance sur trois" de modifier une variable incorrecte). Cela se traduit par $E(k) \leq 1 + (E(k-1) + 2E(k+1))/3$. Par suite, $2E(k+1) - 3E(k) + E(k-1) + 3 \geq 0$, et donc $E(k) - E(k-1) \leq 2(E(k+1) - E(k)) + 3$. Posant $a_i = E(n-i) - E(n-i-1)$, on a bien $a_0 \leq 1$ et $a_{i+1} \leq 2a_i + 3$, soit encore $a_i \leq 2^{i+2} - 3$. Par suite $E(n) \leq E(n-k) + \sum_{i=0}^{k-1}(2^{i+2} - 3) = E(n-k) + (2^{k+2} - 4) - 3k$. En faisant $k = 0$, il vient $E(n) \leq 2^{n+2} - 4 - 3n$. 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 $m \approx 2^{n+3}$...

3. Coupure minimale

i. Soit G un graphe et G' le graphe obtenu après avoir contracté une arête $e = \{u, v\}$. Soit C_0 une coupe minimale de G'. On veut montrer que C_0 est une coupe de G. Soit S'_1 et S'_2 les ensembles de sommets des deux composantes connexes de $G' \setminus C_0$. On suppose sans perte de généralité que le sommet contracté w appartient à S'_1. Toute arête de G reliant un sommet de $S_1 = (S'_1 \setminus w) \cup \{u, v\}$ à un sommet de $S_2 = S'_2$ est une arête de G' reliant un sommet de S'_1 à un sommet de S'_2, donc cette arête est dans la coupe C_0. Par conséquent, C_0 est une coupe de G qui sépare les sommets dans S_1 et les sommets dans S_2.

Toute coupe d'un graphe contracté est une coupe du graphe initial donc les contractions ne diminuent pas la taille de la coupe minimale.

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

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

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 $O(n^4 \log n)$.