Algorithmique avancée

Série d'exercices n° 2 – Algorithmes randomisés
(Correction)

Exercice 1 : Election dans un système anonymes

Dans les systèmes distribués, il y a un modèle appelé *réseau radio sans détecteur de collision* ("*radio networks without collision detection*"). Dans un tel modèle, le temps est discret et à chaque instant t, une station peut décider d'emettre un message ou d'écouter. Un message d'une station s peut être instantanément transmis à tous les voisins et il est correctement reçu par un voisin v de la station (ou processeur) émetrice s si et seulement si s est l'unique voisin de v à lui transmettre un message à cet instant. Dans le cas contraire, v ne distingue pas l'absence totale de message et la collision de plusieurs messages (d'où le terme "sans détecteur de collision").
Un des problèmes fondamentaux dans les systèmes distribués consiste à casser la symétrie d'un système anonyme (toutes les stations sont anonymes) via une élection. Au début de l'algorithme, toutes les stations sont anonymes et à la fin de l'algorithme une unique station est élue (et elle le sait) et les $n-1$ autres stations ne sont pas élues (et elles le savent). On se place dans un cadre ou chaque processeur peut transmettre à tous les autres (graphe sous-jacent complet).

1. Montrer que si le nombre de participants n est connu d'avance par toutes les stations alors il existe un algorithme d'élection en temps moyen constant. Ecrire un tel algorithme.

$\triangleright$ *Pour chaque station en parallèle, avec probabilité $1/n$ envoyer un message. S'il y a une seule émetrice alors cette station est l'élue. Sinon recommencer. Quand n est grand, la probabilité d'avoir une élection est $1/e \sim 0.36$. Donc en moyenne, il faut un temps $O(1)$ pour élire un leader dans un tel système.*

2. Dans le scénario où le nombre total de stations n n'est pas connu, écrire un algorithme d'élection.

$\triangleright$ *On ne peut pas profiter d'une approximation probabiliste de la valeur de n car on ne distingue pas s'il y a trop de message(s) ou s'il n'y en a aucun. Par contre, on peut toujours penser que $2^k \leq n \leq 2^{k+1}$. Une première idée serait de faire des probabilités $1/2^1$, $1/2^2$, $1/2^3$, $\cdots$.*

```
Data : n processeurs anonymes disposés en réseau complet
Result : 1 processeur élu parmi les n
k := 1;
while pas d'élue do
    for i := 1 to k do
        Avec probabilité 1/2^i envoyer un message;
        if seul à envoyer then
            l'élection a lieu on sort de l'algorithme;
            l'unique station à transmettre est l'élue;
            les autres sont non-élues;
        end
    end
    k := k + 1
end
```

Quand on est dans "la bonne tranche", i.e. quand $k \sim \log_2 n$ on a une "bonne probabilité" proche de $1/e$ d'être élue et il nous faut un temps $O(\log^2 n)$ pour y arriver. Pour faire un algorithme en temps $O(\log n)$, il faut remplacer $k := k+1$ par $k := 2.k$.

Exercice 2 : Sous-graphe biparti d'un graphe et MAX-CUT

1. Montrer que dans un graphe avec m arêtes il y a toujours un sous-graphe biparti (colorable avec 2 couleurs) avec au moins $m/2$ arêtes.

$\triangleright$ *On montre par ce résultat avec la méthode probabiliste. Soit $G = (V, E)$ notre graphe avec $|E| = m$. On choisit un sous ensemble T des sommets ($T \subseteq V$) en mettant chaque sommet de V dans T avec probabilité*

1/2. Pour chaque arête $e = (u, v) \in E$, soit X_e la v.a (indicatrice) qui vaut 1 ssi exactement l'un des deux sommets u ou v est dans T (sinon $X_e = 0$. On a

$$\mathbb{E}[X_e] = \mathbb{P}[\{u \in T \text{ et } v \notin T\} \text{ ou } \{u \notin T \text{ et } v \in T\}] = 2.1/4 = 1/2\,.$$

Si X dénombre le nombre d'arêtes avec exactement un sommet dans T alors $\mathbb{E}[X] = \sum_{e\in E}\mathbb{E}[X_e] = m/2$. Pour ce T, on a en moyenne $m/2$ arêtes entre la partie de gauche T et la partie de droite $V\ T$.

Le problème MAXCUT consiste à partitionner les sommets d'un graphe $G = (V, E)$ en deux sous ensembles A et B (i.e. $A \bigcup B = V$ et $A \bigcap B = \emptyset$) telle que le nombre d'arêtes reliant les sommets de A à ceux de B soient maximum. Ce problème est NP-difficile et on veut un algorithme randomisé d'approximation.

2. Montrer la relation entre MAXCUT et les sous-graphes bipartis.

▷ *un sous-graphe biparti avec le maximum d'arêtes entre les sommets rouges et les sommets bleus montre une bipartition qui maximise le nbr d'arêtes entre rouges/bleus.*

3. Elaborer un algorithme FPRAS pour le problème MAXCUT

▷ *On s'inspire de Johnson (cf. transparent numéro 2 du cours) pour élaborer une approximation randomisé qui trouve une coupe d'une certaine taille garantie à un facteur 2 de l'optimale avec grande probabilité.*

- *Trouver une coupe de taille au moins $m/2$ avec le glouton avec proba $2/m$.*
- *En répétant ce glouton un nombre $O(m)$ fois, on garantit avec probabilité $1 - o(1)$ qu'on trouve une coupe de taille au moins $m/2$. Plus précisement, en appliquant l'inégalité de Chebysev :*

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq \lambda] \leq \frac{Var[X]}{\lambda^2}$$

Ici $X = \sum_{i=1}^{T} X_i$ où chaque X_i est un Bernoulli de paramètre $2/m$ (une tentative de trouver plus que $m/2$ qui succède avec proba $2/m$) et T est le temps qu'on laisse l'algo boucler sur le glouton. $\mathbb{E}[X] = \frac{2T}{m}$. $Var[X] = Tpq = 2T/m(1-2/m)$. En choisissant $T = O(m^2)$ et $\lambda = O(m)$ on trouve que la probabilité que l'évènement survienne est d'au moins $1 - O(1/m)$. On peut faire mieux car X suit une loi $Bin(T, 2/m)$ et en utilisant les bornes de Chernoff on a

$$\mathbb{P}[X \geq (1+c)2T/m] \leq \exp\left(-c^2/3.2T/m\right)$$

On fait tourner $T = O(m \log m)$ fois le glouton et on a un algorithme qui trouve une solution 2-approché en temps polynomial avec haute probabilité.

Exercice 3 : 2-SAT est "polynomial"

Dans cet exercice, on va analyser un algorithme randomizé qui s'exécute en temps moyen quadratique pour décider si une formule 2-SAT peut être satisfaite.

1. Rappeler le problème de décision 2-SAT.

▷ *voir cours etc ...*

2. Soit l'algorithme suivant :

```
Data : Une formule 2-SAT construite sur n variables
Result :
Prendre une affectation A aléatoire des variables ;
while A ne satisfait pas la formule do
    Soit C une clause non satisfaite de A;
    Choisir l'une des deux variables et changer son affectation;
end
```

Montrer que sur une instance qui peut être satisfaite cet algorithme renvoie une affectation satisfaisante en un temps moyen $O(n^2)$.

▷ *Soit $S(k)$ = nombre moyen d'itérations nécéssaires jusqu'à ce que les n variables soient **affectées correctement** (par hypothèse la formule peut être SAT) en sachant que k variables sont déjà affectées correctement. Par définition, $S(0)$ est donc la valeur moyenne qu'on cherche. On a*

$$S(k) \leq \begin{cases} \frac{1}{2}\left(S(k+1)+1\right) + \frac{1}{2}\left(S(k+1)+1\right) & \forall k, 0 < k < n \\ 0, \quad k = n & \\ S(1)+1, \quad k = 0 & \end{cases}$$

En effet, si k variables sont déjà correctes, en inversant 1 variable, soit on avance vers $k+1$ variables correctes soit on revient en arrière et on a $k-1$ variables correctes. Dans le pire cas moyen, avec probabilité $1/2$ on avance (ou on revient en arrière). D'où l'inégalité $\leq$. On a $S(n) = 0$ car l'algorithme s'arrête. On a $S(0) = S(1) + 1$. On résout d'abord la récurrence homogène

$$S'(k) = \frac{1}{2}S'(k+1) + \frac{1}{2}S'(k-1)$$

dont l'équation caractéristique est

$$r^k = \frac{1}{2}r^{k+1} + \frac{1}{2}r^{k-1} \text{ ou } r = \frac{1}{2}r^2 + \frac{1}{2}.$$

On trouve $r = 1$ qui est une racine double. Ainsi $S'(k)^h = (c_1 k + c_2)$ pour 2 constantes c_1 et c_2. On cherche maintenant une solution particulière de $S(k)^p = k^2 c$ avec c constante. On développe

$$ck^2 = 0.5c(k+1)^2 + 0.5c(k-1)^2 + 1$$

et on trouve $c = -1$ et donc $S(k)^p = -k^2$. On combine enfin les solutions homogène et particulière

$$S(k) = S'(k)^h + S(k)^p = c_1 k + c_2 - k^2$$

en utilisant les conditions initiales. On trouve $c_1 = 0$, $c_2 = n^2$. Ce qui démontre que $\mathbf{S(0) = (0.0 + n^2) - 0^2 = n^2}$. CQFD