Algorithmes Probabilistes

1. Généralités
2. propriété de Ramsey : la “méthode probabiliste"
3. Couverture par ensembles
4. Programmation linéaire en petite dimension
5. Hachage parfait
6. Routage sur l'hypercube : permutation aléatoire
7. Exclusion mutuelle : les philosophes

Algorithme Probabiliste

Lors de l'exécution, l'algorithme fait des choix probabilistes guidés par des tirages aléatoires : lancers de pièces pile ou face.

Deux catégories

1. Monte Carlo Le résultat du calcul est aléatoire. Toujours rapide, Probablement correct
2. Las Vegas Le temps d'exécution est aléatoire. Toujours correct, Probablement rapide

Outils d'analyse probabiliste

Inégalité de Markov. Soit X une variable aléatoire positive de moyenne $E(X)$. Alors pour tout t :

$$Pr(X \geq t \cdot E(X)) \leq \frac{1}{t}.$$

Inégalité de Chebychev : Soit X une variable aléatoire X de moyenne μ et d'écart-type σ. Alors pour tout a :

$$Pr(|X-\mu| \geq a\sigma) \leq \frac{1}{a^2}.$$

Grandes déviations Soient $X_1, \ldots, X_n$ n variables aléatoires indépendantes valant 1 avec probabilité p et 0 avec probabilité $1-p$, et soit $X = \sum X_i$, $E(X) = np$. Alors :

$$Pr(X \geq (1+\delta)np) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{np}.$$

Analyse probabiliste ou algorithme probabiliste

Analyse de quicksort : Complexité $\Theta(n^2)$ dans le pire cas, mais $O(n \log n)$ en moyenne si les nombres à trier forment une permutation aléatoire.

Quicksort randomisé :

1. Faire une permutation aléatoire des nombres donnés en entrée, puis
2. Exécuter Quicksort

Analyse de quicksort randomisé : La complexité est une variable aléatoire. Quels que soient les nombres donnés en entrée, la complexité est $O(n \log n)$ en moyenne, où la moyenne porte sur le choix de la permutation aléatoire.

Avantages : contrôle de la distribution, donc pas de données pathologiques

Propriété de Ramsey

Dans tout groupe de **n=6** personnes, il existe toujours, ou bien **k=3** personnes qui se connaissent, ou bien **3** personnes qui ne se connaissent pas.

Faux pour les groupes de **n=5** personnes.

Quelle doit être la taille du groupe pour garantir l'existence de **k=4** personnes se connaissant toutes mutuellement ou ne se connaissant pas du tout mutuellement ? **n=17**.

Théorème : Si n est tel que

$$2\binom{n}{k} < 2^{k(k-1)/2},$$

alors il existe des groupes de n personnes sans sous-groupe de k personnes qui se connaissent toutes mutuellement ou ne se connaissent pas du tout mutuellement.

Preuve :

1. On prend une configuration **aléatoire** de n personnes.
   $\Pr\{i \text{ connaît } j\} = 1/2$.
2. On montre que

   $$\Pr\{\exists \text{ sous-groupe de } k \text{ personnes satisfaisant la propriété } \} < 1.$$

3. On en déduit qu'il existe une configuration démontrant le théorème.

Problème ouvert : construction explicite.

Couverture par ensembles

$S_1, \ldots S_m$ ensembles ; Univers $U = \{u_1, \ldots, u_m\}$
Choisir le nombre minimal d'ensembles pour couvrir tous les éléments de U.

Formulation PLNE

$$\min \sum_S x_S$$

$$\forall u, \quad \sum_{S : u \in S} x_S \geq 1$$

$$x_S \in \{0, 1\}$$

Relaxation: $0 \leq x_S \leq 1$.

Arrondi randomisé

Soit $x(S)$ solution fractionnaire optimale. Soit $K = c \log n$.

```
Pour i allant de 1 a K
  C(i) <-- emptyset
  Pour tout S
     Mettre S dans C(i) avec probabilite x(S)
Resultat: C= union des C(i)
```


Analyse

Nombre moyen d'ensemble dans le résultat $= K \sum_S x(S) \leq K \cdot \mathrm{OPT}$.

Probabilité que C soit une couverture ?

Soit a un élément. Probabilité que a ne soit pas couvert par $C(i)$:

$$\Pr\{\forall S \text{ tq } a \in S, S \text{ non choisi }\} = \prod_{S: a \in S} (1 - x(S)).$$

Analyse (suite)

Soit k le nombre d'ensembles contenant a. Sachant $\sum x(S) \geq 1$, l'expression $\prod(1 - x(S))$ est maximisée si les $x(S)$ sont égaux et égaux à $1/k$. Alors:

$$\Pr(a \text{ non couvert dans } C(i)\} \leq (1 - 1/k)^k \leq 1/e$$

$$\Pr(a \text{ non couvert dans } C\} \leq (1/e)^K \leq 1/4n$$

$$\Pr(\exists a, a \text{ non couvert dans } C\} \leq n \cdot 1/4n \leq 1/4$$

$$\Pr\{C \text{ est une couverture }\} \geq 3/4.$$

Programmation linéaire en petite dimension

Problème : n contraintes linéaires (ensemble H), d variables, d petit.

Hypothèse simplificatrices : Sans perte de généralité, on peut supposer que :

- La fonction économique est x_1,
- L'optimum est atteint en un sommet unique du polyèdre, et
- Ce sommet est défini par exactement d contraintes : ensemble S.

Algorithme

Cas de base : si n petit, utiliser l'algorithme du simplexe.

Cas général :

Initialisation : $S \leftarrow \emptyset$

Répéter

Soit R un ensemble **aléatoire** de r contraintes de $H \setminus S$

Résoudre récursivement le problème pour $R \cup S \Rightarrow$ sommet x

Soit V l'ensemble des contraintes violées par x

(Si $|V|$ pas trop grand alors) ajouter V à S

Jusqu'à ce que x satisfasse toutes les contraintes

$r = \min\{d\sqrt{n}, |H \setminus S|\}$; "pas trop grand" : $|V| < 2\sqrt{n}$.

Analyse

- Un passage dans la boucle est **satisfaisant** si $|V|$ n'est pas trop grand.
- A chaque passage satisaisant dans la boucle, S s'enrichit d'au moins une contrainte critique $\Rightarrow$ au plus d passages satisfaisants dans la boucle.
- Chaque passage dans la boucle coûte $O(dn)$ en dehors des appels récursifs
- Les appels récursifs portent sur les ensembles de contraintes de taille au plus $2d\sqrt{n}$.
- Supposons que la probabilité qu'un passage soit satisaisant soit $\geq 1/2$. Alors le nombre moyen total de passages dans la boucle est $\leq 2\sqrt{n}$.

$$T(n) \leq 2dT(2d\sqrt{n}) + O(d^2 n) \Rightarrow T(n) = O(d^2 n + (\log n)^{f(d)}).$$

Probabilité qu'un passage soit satisfaisant

On montre $E(|V|) \leq d(n-r+1)/(r-d) \leq \sqrt{n}$ et on utilise l'inégalité de Markov.

Hachage parfait

Etant donné un univers $U = \{1, 2, \ldots, N\}$ de *clés* et un ensemble X de n clés, on cherche une application injective $X \longrightarrow \{1, 2, \ldots, m\}$ de la forme :

$$h_k(x) = (kx \bmod N) \bmod m,$$

où k est un entier à choisir compris entre 1 et $N-1$

Collisions

A chaque fonction h_k on associe sa mesure de collision : soit $c(i,k)$ le nombre de clés d'image i par h_k. Soit

$$\gamma(k) = \sum_{i=0}^{m-1} c(i,k)^2 \quad - n.$$

Alors h_k est une injection si et seulement si

$$\gamma(k) = 0$$

Lemme : Si N est premier alors on a :

$$\sum_k \gamma(k) \quad \leq \quad \frac{2n(n-1)(N-2)}{m}$$

Conséquence : Si k est tiré au hasard, $E(\gamma(k)) \leq 2n(n-1)/m$, et donc par Markov :

$$Pr(\gamma(k) \leq \frac{4n(n-1)}{m}) \quad > \quad \frac{1}{2}.$$

Algorithme

- `Repeter`
  - Choisir k au hasard dans $1, \ldots N-1$
  - jusqu'à $\gamma(k) < \frac{4n(n-1)}{m}$
- Le temps moyen de calcul est $2n$
- On trouve en moyenne le résultat après deux essais

Routage sur l'hypercube

Dans un réseau représenté par un graphe $G = (X, E)$, chaque sommet i détient un message M_i qu'il doit envoyer à un autre sommet $\sigma(i)$

- Un algorithme de routage consiste à choisir à chaque étape un certain nombres de messages qui vont passer d'un sommet à l'un de ses voisins
- Au cours d'une étape deux messages ne peuvent pas emprunter la même arête
- Il s'agit de minimiser le nombre d'étapes
- On suppose que σ est une permutation (pour simplifier les calculs).

Graphe de l'Hypercube

- Les sommets sont toutes les suites de n bits
- Une arête entre chaque couple de sommets qui diffèrent sur 1 bit

Algorithme déterministe

- A chaque étape un message cherche à se rapprocher de son but en modifiant le bit le plus à gauche du sommet qui diffère de celui du but
- Si deux messages veulent emprunter la même arête on donne priorité à celui qui attend depuis le plus long temps
- Les ex aequo sont départagés au hasard

Algorithme probabiliste

1. **Initialisation** Chaque sommet détermine au hasard une destination intermédiaire $\tau(i)$
2. **Phase 1** On applique l'algorithme déterministe pour aller de chaque i à $\tau(i)$
3. **Phase 2** On applique l'algorithme déterministe pour aller de chaque $\tau(i)$ à $\sigma(i)$

Lemme 1 Deux chemins déterminés par l'algorithme déterministe se rencontrent sur une partie constituée d'arêtes consécutives

Lemme 2 Pour une phase le retard accumulé par chaque message est inférieur ou égal au nombre de messages différents qui empruntent une même arête que lui.

Nombre moyen d'étapes

- $H_{i,j}$ nombre de chemins contenant l'arête (i, j)
- Le nombre total d'arêtes est $Nn = 2^n n$
- Le nombre moyen de chemins empruntant une arête donnée est $1/2$
- La valeur moyenne de la somme des $H_{i,j}$ est $n/2$.
- La probabilité que cette variable dépasse $(\theta + 1)n$ est au plus $2^{-\theta n}$
- Avec probabilité $\frac{2}{N^\theta}$ chaque message atteint sa destination finale en plus de $(2\theta + 4)$ étapes.

Le problème des philosophes

- Des philospohes sont assis autour d'une table, une assiette de spaghetti devant eux
- Chacun partage une forurchette avace son voisin de droite et une autre avec son voisin de gauche
- Il peut, vérifier si une fourchette est libre et s'en saisir, ou en relacher une
- Il mange s'il détient une fourchette dans chaque main

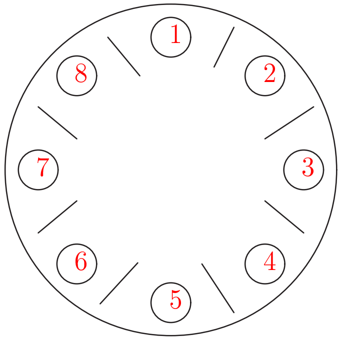

Figure 8: * philosophes autour d'une table

Problème

Trouver un algorithme pour chaque philosophe tel que :

- Deux philosophes ne détiennent pas ensemble la même fourchette Correction
- Chaque philosophe mange au bout d'un temps raisonable vivacité
- Pas de solution symétrique!

Une solution incorrecte

1. Attendre que la fourchette de gauche soit libre, la saisir
2. Attendre que la fourchette de droite soit libre, la saisir
3. Manger
4. Reposer les deux fourchettes

Modélisation

1. Les fourchettes sont des variables partagées $f[i]$ accessibles par i et $i+1$
2. Les états : Repos, Reposer-gauche, Reposer-droite, Critique, Laisser-gauche, Laisser-droite
3. Transition : changement d'état, modification des variables partagées
4. Modèle avec des automates

Algorithme Pair-impair

Pincipe intuitif :

Les philosophes de numéro pair se saisissent d'abord de la fourchette de gauche, ceux de numéro impair de la fourchette de droite

- Les variables partagées sont des files de priorité pouvant contenir deux valeurs
- Un philosophe ne prend une fourchette que s'il est premier sur sa file de priorité
- Une liste d'attente est une suite de philosophes dont chacun d'entre-eux détient une fourchette et attend que celui qui le précède dans la liste repose la sienne
- On vérifie que dans l'algorithme pair/impair, les listes d'attentes sont au plus de taille 2
- Une file de priorité correspond a une fourchette classée première pour deux philosophes

Algorithme probabiliste

Pincipe intuitif :

Chaque philosphe tire au sort la fourchette dont il va se saisir en premier

- Il existe des situations de blocage mais qui interviennent avec une probabilité faible
- Si ℓ est le temps maximum d'exécution d'une étape, on montre que la probabilité pour que l'on arrive en temps 14ℓ à un état dans lequel au moins un philosophe mange est supérieur ou égale à

$$\frac{1}{16}$$

Algorithme

1. **Variables partagées** `f[i]` initialisée à `false` accessible en lecture-écriture par les processus i et $i-1$
2. **Algorithme du processus i :**

```
do forever
• first := random(0,1);
• wait until f[first] = false
• if f[1-first] = false then
  – f[1-first] = true
  – go to Crit
• else f[first] := false
```


Preuve de validité

- On note $\mathcal{T}$ l'ensemble des états où l'un des processus est dans `Try`, et $\mathcal{C}$ l'ensemble des états où l'un des processus est dans la section critique
- On note

$$\mathcal{U} \xrightarrow{t}_p \mathcal{V}$$

  si on passe d'un état de $\mathcal{U}$ à un état de $\mathcal{V}$ en moins de t étapes avec une probabilité supérieure ou égale à p.
- On va montrer :

$$\mathcal{T} \xrightarrow{14\ell}_{1/16} \mathcal{C}$$

Notations

- On note $\mathcal{G}$ l'ensemble des états où l'un des processus a choisi en premier la gauche et où un de ses voisins a fait un premier choix qui n'est pas de la même fourchette.
- On note : $\mathcal{RT}$ l'ensemble des états où tous les processus sont dans un état de `Try`
- On note : $\mathcal{F}$ l'ensemble des états où l'un des processus est dans l'état `Flip`.

Transitions

- On vérifie

$$\mathcal{T} \xrightarrow{3\ell}_1 \mathcal{RT} \cup \mathcal{C}$$

- On a

$$\mathcal{RT} \xrightarrow{4\ell}_1 \mathcal{F} \cup \mathcal{C}$$

- On vérifie

$$\mathcal{F} \xrightarrow{3\ell}_{1/4} \mathcal{G} \cup \mathcal{C}$$

- On vérifie

$$\mathcal{G} \xrightarrow{6\ell}_{1/4} \mathcal{C}$$