

## Algorithmique et probabilités

Un algorithme **probabiliste** est un algorithme qui fait des choix aléatoires au cours de son exécution.

Un algorithme probabiliste fait appel à un ou plusieurs **générateurs aléatoires** tels que:

- tirage à pile ou face (v.a. Bernoulli)
- tirage uniforme d'un entier dans  $\{1, \dots, n\}$
- tirage uniforme d'un réel dans  $[0, 1]$



Pour l'algorithmique probabiliste on suppose qu'on dispose de tels générateurs idéaux. La conception des générateurs pseudo-aléatoires qu'on utilise en pratique est une branche en soit de l'algorithmique, qu'on ne discutera pas ici.

## Cours 7: Algorithmes probabilistes

- Algorithmique et probabilités: outils, méthodes, concepts
- Arrondi aléatoire pour MAXSAT
- Échantillonnage pour les plus proches voisins.
- Existence probabiliste: hachage parfait
- "Randomization" des données pour le routage
- Brisure de symétrie en algorithmique distribuée

## Algorithmique et probabilité

On se concentre sur les aspects **algorithmiques**, et on n'utilisera que quelques outils probabilistes: **l'additivité de l'espérance**

**Inégalité de Markov:** Soit  $X$  une variable aléatoire positive de moyenne  $\mathbb{E}(X)$ : pour tout  $t$

$$\Pr(X \geq t \cdot \mathbb{E}(X)) \leq 1/t$$

**Inégalité de Chebychev:** Soit  $X$  une variable aléatoire de moyenne  $\mu$  et d'écart type  $\sigma$ : pour tout  $a$

$$\Pr(|X - \mu| \geq a \cdot \sigma) \leq 1/a^2$$

**Grandes déviations pour les sommes de Bernoulli:** Soit  $X = \sum_{i=1}^n X_i$  la somme de  $n$  v.a.  $\{0, 1\}$  indépendantes avec  $\Pr(X_i = 1) = p$ . Alors

$$\mathbb{E}(X) = np \text{ et } \Pr(X \geq (1 + \delta)np) \leq e^{-np((1+\delta) \log(1+\delta) - \delta)} \text{ pour } \delta > 0$$

## Des probabilités pour des problèmes déterministes

Les **nombre de Ramsey**: quelle doit être la taille  $n$  d'un groupe de personnes pour garantir qu'il remplit la condition de Ramsey:

contenir  $k$  personnes qui se connaissent toutes 2 à 2, ou  $k$  qui ne se connaissent pas du tout.

- Pour  $k = 3$  on vérifie que 6 personnes suffisent, mais pas 5.
- Pour  $k = 4$  il faut un groupe de  $n = 18$  personnes.

**Théorème:** Soit  $k$  un entier fixé. Si  $n$  vérifie  $2 \binom{n}{k} < 2^{k(k-1)/2}$ ,

il y a un groupe de  $n$  personnes ne satisfaisant pas la propriété de Ramsey.

**Preuve:** On considère une configuration aléatoire de  $n$  personnes où chaque couple de personnes a probabilité  $1/2$  de se connaître.

- On montre que  $\Pr(\exists k \text{ personnes satisfaisant la propriété}) < 1$ .
- On en déduit l'existence. **Pas de construction explicite connue!**

## Arrondi aléatoire pour MAX-SAT

**Donnée.** Un ensemble de clauses  $C_1, \dots, C_m$  sur  $x_1, \dots, x_n$ .

**Modélisation linéaire.** Variables  $y_1, \dots, y_n$  et  $z_1, \dots, z_m$  dans  $\{0, 1\}$ :

$$\text{contrainte associée à } C_j: \quad z_j \leq \sum_{x_i \in C_j} y_i + \sum_{\bar{x}_i \in C_j} (1 - y_i)$$

$$\text{objectif:} \quad \max(z_1 + \dots + z_m)$$

**Solution réelle.** Soit  $\hat{y}_i, \hat{z}_j$  les valeurs des  $y_i$  et  $z_j$  dans un optimum réel du problème linéaire avec  $0 \leq y_i, z_j \leq 1$ .

**Arrondi probabiliste:** Donner à  $x_i$  la valeur VRAI avec proba  $\hat{y}_i$ .

**Analyse:** • L'optimum entier est majoré par l'optimum réel donc le nombre de clauses satisfaisables est au plus  $\hat{z}_1 + \dots + \hat{z}_m$ .

• une clause  $C_j$  à  $k$  variables est satisfaite avec proba au moins

$$(1 - (1 - 1/k)^k) \cdot \hat{z}_j = (1 - 1/e^{-k \log(1-1/k)}) \hat{z}_j \geq (1 - 1/e) \hat{z}_j.$$

• L'espérance du nombre de clauses satisfaites est  $\geq (1 - 1/e) \sum \hat{z}_j$

**Théorème.** L'algo est  $(1 - 1/e)$ -approché en moyenne.

## Algorithmique et probabilité

Analyse probabiliste d'algorithme / algorithme probabiliste:

- **Analyse en moyenne de quicksort:** complexité  $O(n^2)$  dans le pire cas, mais  $O(n \log n)$  en moyenne si les nombres à trier forment une permutation aléatoire.
- **Quicksort randomisé:** appliquer une permutation aléatoire aux entrées avant de trier par quicksort.
- **Analyse probabiliste:** La complexité est une v.a., quelque soit l'entrée la complexité moyenne est  $O(n \log n)$ , où la moyenne porte sur le choix de la permutation effectuée:  
 $\Rightarrow$  plus de données pathologiques...

Algo de type **Las Vegas:** toujours juste, complexité aléatoire (bonne).  
 $\neq$  algo de type **Monte Carlo:** toujours efficace, bonne proba de succès

## Échantillonnage

**Méthode de l'échantillonnage:** on traite le problème à partir d'un sous-ensemble aléatoire des données.

**Premier exemple:** Trouver dans une liste  $a_1, \dots, a_n$  de nombres un  $a_m$  qui soit plus grand que plus de la moitié des  $a_i$ .

• Algo naïf en  $O(n)$ : prendre le max parmi les  $n/2 + 1$  premiers.

• Algo de type Monte Carlo:

choisir  $\sqrt{n}$  éléments au hasard et renvoyer le max

$\Rightarrow$  Probabilité d'échec  $(1/2)^{\sqrt{n}}$ .

## Arrondi aléatoire pour MAX-SAT

**Donnée.** Un ensemble de clauses du type  $(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \dots \vee x_k)$

**Problème.** Trouver une assignation des variables qui maximise le nombre de clauses satisfaites.

**Algo naïf.** Tirer l'assignation au hasard pour chaque variable.

Trivialement cela donne avec grande proba une solution  $\frac{1}{2}$ -approchée.

**Méthode de l'arrondi probabiliste:** se ramener à un problème linéaire, résoudre en réel et arrondir aléatoirement (avec proba bien choisies).

## Randomization pour le routage dans l'hypercube

**Donnée:** Un graphe  $G = (X, E)$  et pour chaque sommet  $i \in X$  un message à envoyer à un destinataire  $\sigma(i)$ . On suppose  $\sigma$  injectif.

**Problème:** Trouver un algorithme de routage qui diffuse les messages en un nombre minimal d'étapes, avec la contrainte qu'un seul message peut circuler par arête à chaque étape.

**Théorème (admis):** Dans un graphe régulier de degré  $d$ , un algo déterministe n'utilisant que la destination d'un message pour choisir son chemin de transmission nécessite  $\Theta(|X|^{1/2}/d)$  étapes.

**L'hypercube.** On se concentre sur le cas de l'hypercube à  $2^n$  sommets:  $X = \{0, \dots, 2^n - 1\}$  et  $(i, j) \in E$  si  $i$  et  $j$  diffèrent sur exactement un bit en binaire.

Pour l'hypercube à  $2^n$  sommets, le degré d'un sommet est  $n$  et la borne inférieure du nombre d'étapes est exponentielle:  $\Theta(2^{n/2}/n)$ .

## Échantillonnage pour MINDIST

**Donnée:**  $S$  un ensemble de  $n$  points du plan

**Problème:** Déterminer  $\delta^* = \min_{x, y \in S} d(x, y)$ .

**Algo naïf** en  $O(n^2)$ , très efficace pour  $n$  petit.

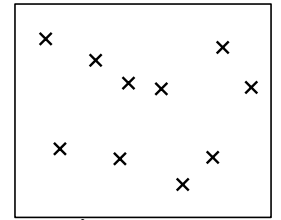
**Algo récursif** en  $O(n \log^2 n)$  asymptotiquement.

**Idée:** diviser en petites régions contenant (on l'espère) peu de points pour y appliquer efficacement l'algo quadratique.

**Théorème d'échantillonnage (admis):** Soit  $S_1 \subset S$  un sous-ensemble formé de  $n^{2/3}$  points pris au hasard uniformément dans  $S$ .

Si on prend  $\delta_1 = \min_{x, y \in S_1} d(x, y)$  alors  $\mathbb{E}(\sum_i n_i^2) = O(n)$ .

**Algorithme:** On itère 2 fois l'échantillonnage pour passer de  $n^{2/3}$  à  $n^{4/9}$  et calculer MINDIST de ces  $n^{4/9}$  pts en  $n^{8/9} = o(n)$  en moyenne.



## Randomization pour le routage dans l'hypercube

**Un algorithme déterministe:** • À chaque étape on tente de modifier le premier bit qui diffère entre l'index du sommet courant et de l'objectif.

• Si deux messages veulent emprunter la même arête on donne priorité au premier arrivé, les ex aequo sont départagés au hasard.

**Variante randomisée:** • Pour chaque sommet on tire au hasard une destination intermédiaire  $\tau(i)$ .

• Les messages sont tous transmis de  $i$  à  $\tau(i)$  par l'algo déterministe (1ère phase), puis de  $\tau(i)$  à  $\sigma(i)$  (2ème phase).

**Lemmes** • Deux chemins de l'algo déterministe ne peuvent se rencontrer dans une phase que le long d'arêtes consécutives.

• Le retard accumulé par un message dans une phase est au plus égal au nombre de messages qui empruntent une même arête que lui.

## Existence probabiliste pour le hachage parfait

**Donnée:** un univers  $U = \{1, 2, \dots, N\}$  de clés et  $X \subset U$  un sous-ensemble de  $n$  clés à traiter.

**Problème:** trouver une application injective  $X \rightarrow \{0, 1, \dots, m-1\}$  de la forme  $h_k(x) = (kx \bmod N) \bmod m$ , où  $k \in \{1, \dots, N-1\}$ .

**Collisions:** soit  $c(i, k)$  le nombre de clés de  $X$  d'image  $i$  par  $h_k$  et  $\gamma(k) = \sum_{i=0}^{m-1} c(i, k)^2 - n$  le nb de  $x \neq y$  avec  $h_k(x) = h_k(y)$

**Lemmes:** •  $h_k$  est une injection ssi  $\gamma(k) = 0$ .

• si  $N$  est premier alors  $\sum_k \gamma(k) \leq 2n(n-1)(N-1)/m$

$\Rightarrow$  si  $k$  est pris au hasard uniformément:  $\mathbb{E}(\gamma(k)) \leq 2n(n-1)/m$

Par Markov on en déduit  $\Pr(\gamma(k) \leq 4n(n-1)/m) > 1/2$ .

• si  $m > 4n^2$ , répéter donne hachage parfait en 2 coups en moyenne.

**Conclusion:** pour avoir espace linéaire on fait un premier hachage avec  $m = 4n$  qui donne des collisions en  $\sqrt{n}$  qu'on hache parfaitement.

## Brisure de symétrie en algo distribuée, les philolophes.

Pas de solution symétrique déterministe évitant le blocage.

### Un algorithme probabiliste:

- Tirer au sort un côté et attendre qu'y soit libre la baguette.
- Une fois prise la première baguette, examiner si l'autre est libre.
- Si oui la prendre, si non reposer la première.

### Remarque:

il est toujours possible que la situation reste bloquée indéfiniment.

**Théorème:** En partant de n'importe quelle configuration, la probabilité pour qu'un philosophe puisse manger avant que ne s'écoule 10 tops est au moins  $1/16$ .

## Randomization pour le routage dans l'hypercube

### Analyse du nombre moyen d'étapes dans la première phase:

Soit  $H_{i,j}$  la v.a. qui vaut 1 si les chemins de  $i$  et  $j$  se croisent, 0 sinon.

Le nombre d'étapes de la 1ère phase est au plus  $n + \max_i \sum_j H_{i,j}$ .

Il y a  $N = 2^n$  chemins de longueur moyenne  $n/2$  pour  $nN$  arêtes, donc le nombre moyen de chemins qui empruntent une arête donnée est au plus  $1/2$ .

⇒ La valeur moyenne des  $\sum_j H_{i,j}$  est au plus  $n/2$ .

**Analyse en probabilité via Markov:** La proba que pour un  $i$  fixé  $\sum_j H_{i,j}$  dépasse  $(1 + \theta)n/2$  est au plus  $2^{-(1+\theta)n}$ .

Avec proba au moins  $1 - 2/N^\theta$  tous les messages atteignent leur destination intermédiaire en moins de  $(3 + \theta)n/2$  étapes.

⇒ Algo de type **Las-Vegas**, presque toujours efficace.

## Brisure de symétrie en algo distribuée, les philolophes.

Dans les problèmes d'algorithmique distribuée, plusieurs processus exécutent un code identique, ce qui conduit souvent à des situations de blocage pour cause de symétrie.

**Illustration: le diner des philosophes.** Des philosophes sont assis autour d'une table pour diner. Devant chaque convive une assiette de nouilles chinoises, entre chaque assiette **une** baguette.

- À chaque top d'horloge un philosophe peut se saisir d'une baguette libre ou en relacher une.
- Dès qu'un philosophe a 2 baguettes dans les mains il peut manger.

### Exemple de processus conduisant à un blocage:

- attendre que la baguette de gauche soit libre et la saisir
- attendre que la baguette de droite soit libre et la saisir
- manger et reposer les baguettes.