Modèles de calcul, Complexité, Approximation et Heuristiques

# Modèle probabiliste: Algorithmes et Complexité

Jean-Louis Roch

Master-2 Mathématique – Informatique

Grenoble-INP – UJF

Grenoble University, France

# L'aléatoire est-il utile ?

$\hookrightarrow$ Complexité interactive, classe IP

- Graphes Isomorphisme $\in$ NP. Mais preuve du NON (ie GNI) ? ?
- IP = système de *preuve interactive* :
  - $V$ = vérifieur probabiliste en temps polynomial $P$= prouveur dynamique

$$\begin{cases} x \in L \Longrightarrow \exists \text{prouveur honnête} P, \Pr[V(x,P) \text{accepte}] = 1 \\ x \notin L \Longrightarrow \forall \text{prouveurs} P', \Pr[V(x,P') \text{accepte}] \leq \frac{1}{2}. \end{cases}$$

  - Exemples : GNI, #3-SAT $\in$ IP.
  - Théorème : IP = PSPACE

# PCP et NP

- $P \subset (ZPP = NP \cap co\text{-}NP) \subset RP \subset NP$.
- PCP : Vérifieur polynomial probabiliste (prouveur statique)

$$\begin{cases} x \in L \Longrightarrow \exists \text{preuve} \Pi, \Pr[V(x, \Pi) \text{accepte}] = 1 \\ x \notin L \Longrightarrow \forall \text{preuves} \Pi, \Pr[V(x, \Pi) \text{accepte}] \leq \frac{1}{2}. \end{cases}$$

  $PCP[r(n), q(n)]$ = langages qui admettent une vérification probabiliste avec $O(r(n))$ bits aléatoires et $O(q(n))$ bits consultés dans la (présupposée) preuve, qui est vérifiée en temps $t(n) = n^{0(1)}$.
- Un prouveur IP est plus puissant qu'un prouveur statique contraint qu'un ppour (Plus grand que IP)
  - Théorèmes directs : P = PCP[0,0], NP = PCP $[0, n^{O(1)}]$, co-RP=PCP$[n^{O(1)}, 0]$
  - Théorème : NP = $PCP[O(\log n), O(1)]$.

# Dérandomisation

*"Dérandomisation"* = élimination de l'aléatoire pour obtenir un algorithme déterministe.

Plan du cours :

- Classes de complexité probabilistes
- Dérandomisation et complexité
- Exemple 1 : MAX-CUT
- Exemple 2 : par évaluation/interpolation [Schwartz-Zippel]
- Optimisation inconsciente : work-stealing distribué

# Introduction

Deux idées récurrentes en algorithmique probabiliste :

- Au moins une valeur d'une v.a. est plus grande que sa moyenne.
- Si un objet choisi aléatoirement dans un univers satisfait une propriété avec une probabilité non nulle, alors au moins un objet satisfait cette propriété.

# Dérandomisation et complexité

## Dérandomisation brutale

- Un langage $L \in RP$ ssi il existe un algorithme déterministe $A$ de temps $t(.)$ polynomial et un polynôme $r(.)$ tel que $\forall x$ :
  $x \in L \Longrightarrow \Pr_{r \in \Sigma^{r(n)}} [A(x, r) \text{accepts}] = 1.$
  $x \notin L \Longrightarrow \Pr_{r \in \Sigma^{r(n)}} [A(x, r) \text{accepts}] \leq \frac{1}{2}$ ;
  $|x| = n$, $A$ utilise au plus $r(n)$ bits aléatoires et a un temps borné par $t(n)$.
- Dérandomisation brutale : exécuter $A$ sur toutes les séquences de $r(n)$ bits et élire la majorité : temps $= 2^{r(n)} t(n)$.

## Dérandomisation non-uniforme

Théorème [Adleman] *Tout langage (ie fonction booléenne) qui peut être calculé par une famille de circuits probabilistes de taille polynomiale peut être calculé par une famille de circuits déterministes de taille polynomiale.*

Preuve : $\forall x$ de taille $n$, construction d'un circuit de taille $\leq (n+1)t(n)$.

ATTENTION : **non uniforme** : ne dit rien sur $RP \subset ? P$ !

- soit $C_n(x, r)$ un circuit de taille $t(n) = n^{O(1)}$ qui prend en entrée $x \in \{0,1\}^n$ et $m$ bits aléatoires $r$.
  - Si $x \notin L$, $C_n(x, r) = 0$ : le circuit donne la bonne réponse.
  - sinon $\mathbb{E}_{ri \in \{0,1\}^m} C_n(x, r) = \frac{1}{2}$.
- soit $M$ la matrice de taille $|L| \times 2^m$ définie par $M(x, r) = C_n(x, r)$ :
  - chaque ligne de $M$ correspond à un élément $x$ de $L$ ;
  - chaque colonne de $M$ correspond à un aléa $x$ de $m$ bits.
- Dans $M$, chacune des $|L|$ lignes contient au moins $2^m/2$ coef =1. Donc $\mathbb{E}\left[\sum_{r \in \{0,1\}^m} \sum_{x \in L} M(x, r)\right] \geq 2^{m-1}.|L|$ ; donc il existe au moins une colonne dont la somme des coefficients est supérieure à $\frac{|L|}{2}$ qui est la moyenne des sommes des colonnes. Soit $r_1$ l'indice de cette colonne.
- Soit $C^{DET}_{n,r_1}(.)$ le circuit $C_n(., r_1)$ : il calcule la valeur de $f(x)$ pour au moins la moitié des valeurs $x$ : on supprime donc les lignes correspondantes et il reste au plus $2^{n-1}$ lignes dans $M$. !
- Pour $i = 2..log_2 n + 1$, réitérer pour construire $C^{DET}_{n,r_i}(.)$ = le circuit $C_n(., r_i)$ : à chaque étape, on élimine au moins la moitié des $x$ restants.
- Finalement, Le circuit $\mathrm{OR}\left(C^{DET}_{n,r_1}(.), \ldots, C^{DET}_{n,r_{n+1}}(.)\right)$ calcule $L$ en temps déterministe $nt(n) = n^{O(1)}$.

## Dérandomisation uniforme

- Stratégie générale par générateur pseudo aléatoire : approcher les aléas $r(n)$ par des aléas $l(n)$ plus courts :

$$\forall x : \Pr_{r \in \Sigma^{r(n)}} [A(x, r)\text{accepts}] \simeq_{\pm n^{-2}} \Pr_{s \in \Sigma^{l(n)}} [A(x, f(s))\text{accepts}]$$

  avec $f : \Sigma^{l(n)} \longrightarrow \Sigma^{r(n)}$ calculée en temps $t_f(n)$.
- Alors $L$ peut être décidé en temps $O\left(2^{l(n)} t(n) t_f(n)\right)$.
- Si $l(n) = O(\log n)$, on obtient une dérandomisation polynomiale.

## Dérandomisation uniforme

- Définition : Soit $U_m$ la distribution uniforme sur $\{0,1\}^m$. Une distribution $R$ sur $\{0,1\}^m$ est $(S,\epsilon)$-pseudo-aléatoire ssi $\forall C$ circuit de taille $S$ :

$$|\Pr[C(R)=1]-\Pr[C(U_m)=1]|<\epsilon.$$

  Soit $S:\mathbb{N}\to\mathbb{N}$ croissante ; un générateur $G:\{0,1\}^\star\to\{0,1\}^\star$ calculable en temps exponentiel $2^n$ est $S(l)$-pseudo-aléatoire si $|G(z)|=S(|z|)\forall z\in\{0,1\}^\star$ et si pour tout $l\in\mathbb{N}$ la distribution $G(U_l)$ est $\left(S(l)^3,10^{-1}\right)$-pseudo-aléatoire.
- Théorème : Si il existe un générateur $2^{l\times\epsilon}$-pseudo-aléatoire avec $\epsilon>0$, alors BPP=P.

# De glouton probabiliste à glouton déterministe

## Algorithme probabiliste d'otimisation

- Pour un problème d'optimisation (exemple MAX $f$),
- un algorithme probabiliste glouton fait des choix aléatoires successifs $r_1, \ldots, r_n$ pour retourner une solution approchée $\tilde{f}(r_1, \ldots, r_n)$ de MAX $f$.

## Méthode

- on remplace le choix de $r_i$ par la valeur déterministe $a_i$ qui MAXimise l'**espérance** de $\tilde{f}(r_1, \ldots, r_n)$ sachant $a_1 = r_1, \ldots, r_i = a_i$.
- Soit $a_i$ tel que $E_{r_{i+1},\ldots r_n}[\tilde{f}(a_1, \ldots a_i, r_{i+1}, \ldots, r_n)] = \max_x E_{r_{i+1},\ldots r_n}[\tilde{f}(a_1, \ldots a_{i-1}, x, r_{i+1}, \ldots, r_n)]$.

# MAX-CUT

## Problème MAX-CUT

- Entrée : un graphe $G = (V, E)$. On note $n = |V|$ et $m = |E|$.
- Sortie : une partition $V_1, V_2$ de $V$ qui maximise la taille de la coupe,
  i.e. le nombre d'arêtes entre $V_1$ et $V_2$.

Le problème de décision associé à MAX-CUT est NP-complet.

## Idée pour un algorithme probabiliste

Soit $A, B$ une partition arbitraire de $V$ et soit $e \in E$ ; on a 3 cas :

- Soit $e$ a ses deux extrémités dans $A$ ;
- soit $e$ a ses deux extrémités dans $B$ ;
- soit $e$ a une extrémité dans $A$ et l'autre dans $B$,
  autrement dit $e \in \text{coupe}(A, B)$.

# Algorithme probabiliste pour MAX-CUT

## MAX-CUT Monte Carlo

$V_1 = \emptyset$ ; $V_2 = \emptyset$ ;
for $i = 1, \ldots, n$
    $b =$ rand(1,2) ;
    Mettre le sommet $v_i$ dans $V_b$.

## Analyse de performance

$$\mathbb{E}[\text{taille de la coupe}] \geq \frac{m}{2}.$$

Preuve : pour tout $e \in E$, la probabilité que $e$ soit dans la coupe est $\frac{1}{2}$. Par linéarité de l'espérance,
$\mathbb{E}[\text{taille de la coupe}] = \mathbb{E}[\sum_{e \in E} \mathbf{1}_{e \in \text{coupe}}] = \sum_{e \in E} \mathbb{E}[\mathbf{1}_{e \in \text{coupe}}] = \frac{m}{2}$.

# Elimination de l'aléatoire

## *Acceptation conditionnelle*

Principe : éliminer les choix pour maximiser l'espérance...

- Supposons qu'on a déjà placé $v_1, \ldots, v_i$ dans $V_1$ et $V_2$ de manière déterministe.
- Quelle est alors l'espérance de la taille de la coupe si on choisit $v_{i+1}, \ldots, v_n$ selon l'algorithme probabiliste ?
  Soit $E_1 = E[\text{coupe} | v_1, \ldots, v_i \text{ placés et } v_{i+1} \in V_1]$ idem $E_2$) :
  $$\text{E}[\text{coupe} | v_1, \ldots, v_i \text{ placés}] = \frac{1}{2}E_1 + \frac{1}{2}E_2$$

## Algorithme dérandomisé

- for $i = 1, \ldots, n$
  Calculer $E_1(i)$ et $E_2(i)$ ;
  Si $E_1 > E_2$ mettre $v_{i+1}$ dans $V_1$, sinon dans $V_2$.
- Approximation déterministe de ratio 2.

car $\frac{m}{2} = E[\text{coupe}] \leq E[\text{coupe} | v_1 \text{ placé}] \leq E[\text{coupe} | v_1, v_2 \text{ placés}]$

# Comment calculer la plus grande espérance ?

- On suppose placés $v_1, \ldots, v_i$ dans $V_1$ et $V_2$ ; on doit choisir $v_{i+1}$.
- Soit à cet instant : $c$ la taille de la coupe,
  $d$ le nombre d'arêtes dans $V_1$ et $V_2$,
  $m_1 =$ le nombre d'arêtes entre $v_{i+1}$ et $V_1$,
  $m_2 =$ nombre d'arêtes entre $v_{i+1}$ et $V_2$.
- Il reste $r = m - c - d - m_1 - m_2$ arêtes d'extrémités dans $\{v_{i+2}, \ldots, v_n\}$.
- $E_1 = E[\text{coupe} | v_1, \ldots, v_i \text{ placés et } v_{i+1} \in V_1] = c + m_2 + \frac{r}{2}$
  et $E_2 = E[\text{coupe} | v_1, \ldots, v_i \text{ placés et } v_{i+1} \in V_2] = c + m_1 + \frac{r}{2}$.
- Finalement le test $(E_1 > E_2) \Longleftrightarrow (m_2 > m_1)$.
- On obtient un algorithme déterministe, glouton.

# Autres exemples

- MAX-3-LIN
- MAX-SAT
- Coloriage d'hypergraphe (discrepancy)
- ...

# Test déterministe de nullité d'un polynôme

- Problème ZEROP : test de nullité d'un polynôme : ZEROP $\in$ co-RP
- Soit $P \in \mathbb{F}[X_1, \ldots, X_n]$ polynôme de degré total $d$ sur un corps $\mathbb{F}$ ;
- Schwartz-Zipppel : en choisissant au hasard un vecteur $r \in I^n$ avec $I$ grand, si l'évaluation $P(r) = 0$, on décide $P = 0$ (avec probabilité d'erreur $\leq \frac{d}{|I|}$).
- Si on peut évaluer $P$ avec un circuit arithmétique de taille $m$, alors $\exists m^2$ vecteurs $r_1, \ldots, r_{m^2}$ tels que si $P \neq 0$, alors $\exists i : P(r_i) \neq 0$.
- Si on est capable de construire $r_1, \ldots, r_m$, on construit ainsi un algorithme déterministe pour décider si $P = 0$.
- Problème : bien que paraissant facile si $I$ grand (*finding a hay in a haystack)* on ne sait pas le faire dans le cas général en temps polynomial déterministe
  (et cela entrainerait des bornes inférieures non triviales, comme [soit NEXP $\not\subset P_{/Poly}$ ou perm $\notin AlgP^{\mathbb{Z}}_{/Poly}$] )