**INF 581: Du calcul probabiliste au calcul quantique** **Hiver 2011**

# Cours 1 — 3 Janvier

*Enseignant : Frédéric Magniez* *Rédacteur : Julien Faure*

Ce cours a pour but la définition de différents modèles et la présentation de quelques exemples d'algorithmes probabilistes.

## 1.1 Modèle

On considère accessible des ressources aléatoires $r \in \{0,1\}^*$.
On pourra par la suite considérer $r \in [|a,b|]^*$ où a et b sont des entiers.

Soit $f\ :\ X \to Y$ et $A$ un algorithme.

**Définition 1.1.** *On dit que $A$ calcule $f$ sans erreur avec une complexité moyenne $T$ si et seulement si :*

1. $\forall x \in X,\ A(x)$ *renvoie* $f(x)$
2. $\forall x \in X,\ \underset{r\in\{0,1\}^*}{E}(C(A(x,r))) \leq T(|\ x\ |)$ *où* $|\ x\ |$ *est la taille de l'entrée et* $C(A(x,r))$*la complexité de A sur l'entrée $x$ avec les choix aléatoires $r$.*

*On étudie alors la complexité moyenne (average case).*

**Définition 1.2.** *On dit que $A$ calcule $f$ sans erreur avec une probabilité d'échec $\delta$ et une complexité $T$ si et seulement si :*

1. $\forall x,\ \forall r,\ si\ A(x,r)\ n'échoue\,pas,\ alors\ A(x,r)\,renvoie\ f(x)$
2. $\forall x,\ Pr(\ A(x,r)\ \textit{échoue}\ )\ \leq \delta$
3. $\forall x,\ \forall r,\ C(A(x,r)) \leq T$

*On étudie alors la complexité dans le pire cas (worst case).*

En général, on choisit $\delta = 1/2$. Pour passer de $\delta_0 = 1/2$ à $\delta_k = 1/2^k$on réitère au plus k fois le même algorithme avec de nouveaux bits aléatoires. On a de manière immédiate : $T\left(\delta = 1/2^k\right) \leq k\,T\,(\delta = 1/2)$.

On considère maintenant des langages : $Y = \{0,1\}^*$ et $L = \{x \mid f(x) = 1\}$.

**Définition 1.3.** *One sided error (RP)*
*A calcule $f$ avec une one-sided error $\delta$ et une complexité $T$ si et seulement si :*

1. $\forall x,\ \forall r,\ C(A(x,r)) \leq T$
2. *si $x \in L$, alors $Pr(A(x,r)\ accepte) = 1$*

3. *si* $x \notin L$*, alors* $Pr\left(A\left(x,r\right)\ accepte\right) \leq \delta$

En général, on choisit $\delta = 1/2$. (il suffit d'avoir $\delta < 1$).
Pour ramener l'erreur à $1/2^k$, on relance l'algorithme $k$ fois et on accepte si et seulement si les k exécutions ont toutes été acceptées.

**Définition 1.4.** *Co-RP*
*A est dans co-RP si et seulement si :*

1. $\forall x,\ \forall r,\ C\left(A\left(x,r\right)\right) \leq T$
2. *si* $x \in L$*, alors* $Pr\left(A\left(x,r\right)\ accepte\right) \geq 1-\delta$
3. *si* $x \notin L$*, alors* $Pr\left(A\left(x,r\right)\ accepte\right) = 0$

**Définition 1.5.** *Two-sided error (BPP)*
*A calcule* $f$ *avec une two-sided error* $\delta$ *et une complexité* $T$ *si et seulement si :*

1. $\forall x,\ \forall r,\ C\left(A\left(x,r\right)\right) \leq T$
2. *si* $x \in L$*, alors* $Pr\left(A\left(x,r\right)\ accepte\right) \geq 1-\delta$
3. *si* $x \notin L$*, alors* $Pr\left(A\left(x,r\right)\ accepte\right) \leq \delta$

Il faut $\delta < \frac{1}{2}$. On prendra généralement $\delta = \frac{1}{3}$.
Dans ce dernier cas on acceptera le résultat si la majorité des executions l'ont accepté.

## 1.2 Premiers exemples

### Exemple 1 : Primalité

$L = \{p | p\ nombre\ premier\}$, $x \in \mathbb{N}^*$, $|x| = log_2(x)$.

Le crible d'Eratosthène nous donne un résultat en $\sqrt{n}$ ce qui est trop long.

**Théorème 1.6.** *Rappel : Petit théorème de Fermat*
*Si p est premier,* $\forall a \in \{1, 2, ..., p-1\}\ \ a^{p-1} \equiv 1\,[p]$

**Algorithme :**

1. On tire a au hasard, uniformément dans $\{1, 2, ..., p-1\}$.
2. Si $a \wedge p \neq 1$ on rejette le résultat.
3. Si $a^{p-1} \neq 1\,[p]$ on rejette le résultat.
4. Sinon on accepte le résultat.

La complexité de cet algorithme est logarithmique en nombre d'opérations arithmétiques.

**Théorème 1.7.** *L'algorithme précédent est de type One-sided error.*

**Preuve:**
- Si p est premier, l'algorithme accepte tout le temps (petit théorème de Fermat).
- On rapelle que p, non premier, est de Carmichael si pour tout a premier avec p on a $a^{p-1} \equiv 1\,[p]$.
  Supposons que p ne soit pas un nombre de Carmichael.
  Soit $a_0$ tel que $a_0 \in \{0, 1..., p-1\}$ , $a_0 \wedge p = 1$ et ${a_0}^{p-1} \neq 1\,[p]$.

  **Lemme 1.8.** $Pr_{a \in \{0,1...,p-1\}\ et\ a \wedge p = 1}\left[a^{p-1} \equiv 1\,[p]\right] \leq \frac{1}{2}$

  **Preuve:** Soit G le groupe multiplicatif tel que $G = \{a | a \wedge p = 1\}$ et H le sous-groupe de G tel que $H = \{a \in G | a^{p-1} \equiv 1\,[p]\}$.
  $a_0 \notin H$ et $a_0 \in G$ donc H est un sous-groupe stricte de G et donc $|H| \leq \frac{|G|}{2}$.
  □

  Le lemme implique que si p, non-premier, n'est pas de Carmichael on a :
  $Pr_a\left[l'\ algorithme\ rejette\ en\ (3)\ sachant\ qu'\ il\ a\ accepté\ en\ (2)\right] \geq \frac{1}{2}$

  On rapelle que $Pr(A \cap B) = Pr(B)Pr(A|B)$.

  $Pr_a\left[l'\ algorithme\ rejette\ en\ (2)\ ou\ en\ (3)\right] = \Delta$
  $= Pr_a\left[l'\ algorithme\ rejette\ en\ (2)\right] + Pr_a\left[l'\ algorithme\ rejette\ en\ (3)\ et\ pas\ en\ (2)\right]$
  $= \Delta + (1-\Delta)Pr_a\left[l'\ algorithme\ rejette\ en\ (3) | l'\ algorithme\ accepte\ en\ (2)\right]$
  $\geq \Delta + \frac{1-\Delta}{2} = \frac{1+\Delta}{2} \geq \frac{1}{2}$

  p est maintenant un nombre de Carmichael
  $p - 1 = 2^t u$
  On désire calculer $a^{p-1}\,[p]$. Soit i le dernier entier tel que $v = a^{2^i u} \neq 1\,[p]$
  $v^2 = 1 = a^{2^{i+1}u}$.(si p premier on a $v = -1$)

  **Théorème 1.9.** $Pr_a\left[v = -1\ ou\ i\ n'\ existe\ pas\right] \leq \frac{1}{4}$

  Pour plus de détails, voir le papier de Miller et Robin.
  □

## Exemple 2 : Test de multiplication de matrices

On considère des matrices de $M_n(\mathbb{Z}/2\mathbb{Z})$. $L = \{(A, B, C) | AB = C\}$.

### Problème :

On cherche un algorithme prenant en entrée trois matrices A,B et C et renvoyant en sortie ACCEPTER si le triplet est dans L, REJETER sinon.

### Algorithme (Freivald) :

1. Choisir $r \in \{0, 1\}^n$.

2. Calculer $u = Cr$.
3. Calculer $v = Br$.
4. Calculer $w = Av$.
5. Renvoyer ACCEPTER si $u = w$, REJETER sinon.

**Théorème 1.10.** *L'algorithme précédent est de type One-sided error, de complexité* $\theta(n^2)$.

**Preuve:**
- La complexité est de manière évidente en $\theta(n^2)$.
- Si $AB = C$ l'algorithme renvoie bien ACCEPTER.
- On considère maintenant $(A, B, C) \notin L$.
  Soit $r_0$ tel que $ABr_0 \neq Cr_0$. $D = AB - C$
  $H = \{r | Dr = 0\}$ est un sous espace vectoriel de $\{0,1\}^n$ strict car $r_0 \notin H$.
  On a donc $\frac{|H|}{|\{0,1\}^n|} \leq \frac{1}{2}$, ce qu'il fallait démontrer.

$\square$

**Théorème 1.11.** *L'algorithme précédent reste valable sur tout anneau.*

**Preuve:** Le début de la preuve précédente reste valable.
Soit $d_{i_0 j_0}$ un coefficient non nul de D. $Dr = (\sum_j (d_{ij} r_j))$
$\forall r,\ (Dr)_{i_0} = \sum_{i=1}^{n} d_{i_0 j} r_j = d_{i_0 j_0} r_{j_0} + \sum_{j \neq j_0} d_{i_0 j} r_j$
On fixe $(r_1, r_2, ...r_{j_0-1}, r_{j_0+1}..., r_n)$.
$r^0 = (r_1, r_2, ...r_{j_0-1}, 0, r_{j_0+1}..., r_n)$ et $r^1 = (r_1, r_2, ...r_{j_0-1}, 1, r_{j_0+1}..., r_n)$.
Comme $d_{i_0 j_0} \neq 0$ on a $(Dr^0)_{i_0} \neq (Dr^1)_{i_0}$ ce qui suffit pour conclure (au moins un r sur deux n'annule pas D).

$\square$

## Exemple 3 : Test de commutativité

### Problème :

Ayant en entrée un groupe fini $G$ et n éléments $h_1, h_2...h_n$ de $G$, on désire trouver un algorithme qui renvoie ACCEPTER si $\forall (i,j),\ h_i h_j = h_j h_i$ et REJETER sinon.

On s'interesse à la complexité en nombre d'opérations de groupes (et non pas au nombre d'additions, multiplications... bien que G puisse être $M_n$ par exemple).
On rapelle que le centre du groupe $G$ est défini par : $Z(G) = \{k \in G | \forall g \in G,\ kg = gk\}$ et on appelle $H$ le sous-groupe généré par les $h_i$.

On ne sait pas calculer H de manière efficace.

### Hypothèse :

On suppose que l'on sait générer uniformement au hasard un élément de $H$.

**Algorithme :**

- Générer h et k uniformément au hasard dans $H$.
- Renvoyer ACCEPTER si $hk = kh$.
- Renvoyer REJETER sinon.

**Théorème 1.12.** *L'algorithme précédent est de type One-sided error de complexité 2 opérations de groupe et 2 échantillonnages dans $H$.*

**Preuve:**
- Le calcul de la complexité est immédiat.
- Si les $h_i$ commutent l'algorithme renvoie bien ACCEPTER.
- Supposons maintenant H non commutatif. $Z(H) \neq H$ et $H$ est un sous-groupe donc $Pr_{h \in H}[h \in Z(H)] \leq \frac{1}{2}$.
  On note $L_h = \{g \in H | gh = hg\}$. On a $L_h \neq H$ si $h \notin Z(H)$.
  Soit $h \notin Z(H)$.
  Comme $L_h$ est alors un sous-groupe stricte de $H$ on a : $Pr_{k \in H}[k \in L_h] \leq \frac{1}{2}$.
  Finalement, $Pr_{k,h}[kh \neq hk] \geq Pr[h \notin Z(H) \; et \; k \notin L_h] = Pr[k \in L_h | h \notin Z(H)] \times Pr[h \notin Z(H)] \geq \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$

□

**Problème de la génération d'éléments dans H :**

La génération uniforme au hasard d'éléments de $H$ n'est pas évidente. Cependant, pour que le résultat demeure il suffit d'avoir un générateur d'éléments de $H$ tel que si $K$ est un sous groupe stricte de H quelconque, $Pr_{h \text{ issu de notre générateur}}[h \in K] \leq \frac{1}{2}$ $(\diamond)$

**Générateur :**

- Choisir uniformément au hasard $r \in \{0,1\}^n$.
- Calculer $h = {h_1}^{r_1} {h_2}^{r_2} \ldots {h_n}^{r_n}$
- Renvoyer h.

**Lemme 1.13.** *Notre générateur satisfait* $(\diamond)$.

**Preuve:** K un sous-groupe stricte de H.
Soit $i_0$ tel que $h_{i_0} \notin K$. $r \in \{0,1\}^n$.
$h = {h_1}^{r_1} {h_2}^{r_2} \ldots {h_{i_0}}^{r_{i_0}} \ldots {h_n}^{r_n}$
On fixe les $r_i$ pour $i \neq i_0$. On note $h^0$ le produit pour $r_{i_0} = 0$ et $h^1$ celui pour $r_{i_0} = 1$.
$h^0 = ab$ et $h^1 = a h_{i_0} b$.
On prend $i_0 = min\{i | h_i \notin K\}$ de telle sorte que a soit dans $K$. Si $b \in K$ alors $h^0 \in K$ mais $h^1 \notin K$ car sinon on aurait $a^{-1} h^1 b^{-1} = h_{i_0} \in K$ ce qui par hypothèse est faux.
Si $b \notin K$ on a directement $h^0 \notin K$. On voit donc qu'au moins l'un de ces deux éléments n'est pas dans $K$ ce qui nous suffit pour conclure. □

## 1.3 Couplages parfaits

On considère A et B deux ensembles de même cardinal n.
$G$ est un graphe d'arêtes $E \subseteq A \times B$.

**Définition 1.14.** *Un couplage est un ensemble $M \subseteq E$ tel que $\forall e, e' \in M,\ e \cap e' = \emptyset$.*
*Il est dit parfait si $|M| = n$.*

**Théorème 1.15.** *Il existe un algorithme deterministe qui trouve un couplage parfait (si un tel couplage existe) et de complexité $\theta(n^{\frac{5}{2}})$.*

**Théorème 1.16.** *Il existe un algorithme probabiliste de type One-sided error de complexité $\theta(n^{\omega})$, avec $\omega < 2.38$. Ce coût correspond au calcul d'un déterminant.*

**Preuve:** Première partie.
$A_G = (a_{ij})_{i \in A,\, j \in B}$ avec $a_{ij} = 0$ si $(i,j) \in E$ et $a_{ij} = X_{ij}$ sinon (matrice de Frobenius).
$Det(A_G) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} a_{i\sigma(i)}$
$PERM(A_G) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)}$ □

**Lemme 1.17.** *$Det(A_G) \neq 0$ (i.e les coefficients du polynôme ne sont pas tous nuls) si et seulement si il existe un couplage parfait dans G.*

**Remarque :**

$PERM(A_G)(1, 1, ..., 1)$ est le nombre de couplages parfaits de G.

**Algorithme :**

Soit p premier, tel que $n^2 < p < 2n^2$.
- Choisir uniformément au hasard $a_{ij} \in \{0, 1, ...p-1\}$.
- Calculer $d = Det(A_G)(a_{ij})$ modulo p, en substituant $X_{ij}$ par $a_{ij}$ dans $A_G$.
- Renvoyer ACCEPTER si $d \neq 0\,[p]$.
- Renvoyer REJETER sinon.

**Preuve:** Deuxième partie.

- La complexité correspond bien à un calcul de déterminant.
- D'après le lemme, si il n'existe pas de couplage parfait, l'algorithme renvoie toujours REJETER.
- Supposons qu'il existe un couplage parfait.
  On va montrer que dans ce cas l'algorithme renvoie ACCEPTER avec une probabilité supérieure à $1 - \frac{1}{n}$.
  Tout dabord, remarquons que $Det(A_G)$ est un polynôme de degré inférieur à n, à $n^2$ variables (au plus).
  On cherche $Pr\,[Det(A_G)(a_{ij}) = 0]$. D'après le lemme de Schwartz-Zippel présenté par la suite, cette probabilité est inférieure à $\frac{n}{p}$.

**Lemme 1.18.** *Lemme de Schwartz-Zippel*
*$f$ un polynôme à $m$ variables de degré $d$ sur un corps $F$.*
*Si $E$ est un polynôme non nul alors $Pr_{a_1,..a_m \in F}\left[f(a_1,...a_m) = 0\right] \leq \frac{d}{|F|}$.*

**Preuve:** Nous allons monter que le nombre de racines de $f$ est inférieur ou égal à $d|F|^{m-1}$.

- Si $m = 1$ le résultat est vrai (un polynôme à une variable, de degré $d$, a au plus $d$ racines).
- Considérons maintenant $m \geq 2$. $1 \leq d \leq |F|$.
  On décompose $f$ en $g + h$ où $g$ est homogène de degré $d$ (non identiquement nul) et $h$ est de degré inférieur stricte à $d$.
  Soit $y \in F^m$, $y \neq (0, 0...0)$ tel que $g(y) \neq 0$. Pour $x \in F^m$ on définit $L_x = \{x + ty | t_i nF\}$.
  $L_x = L_{x'}$ si $x - x' = ky$, $k \in F$ et $L_x \cap L_{x'} = \emptyset$ sinon. Le nombre de (lignes) $L_x$ distinctes est $|F^{m-1}|$ (pour chaque ligne on a $|F|$ représentants).
  Prenons une ligne $L_x$.
  $l(t) = f(x + ty) = g(x + ty) + h(x + ty)$ (polynôme de degré inférieur à d).
  En remarquant que le coefficient de $t^d$ dans $g(x + ty)$ est $g(y)$ et en appliquant le lemme pour $m = 1$ on montre que $f$ a au plus $d$ racines sur $L_x$ ce qui conclut le preuve.

$\square$

$\square$