Algorithmes probabilistes.

Dans ce chapitre, on s'intéresse aux algorithmes probabilistes et les classes de complexité correspondantes. Le modèle de calcul sera les *machines de Turing probabilistes*, qui ressemblent aux machines de Turing non-déterministes, mais pas exactement. Plus précisément, on aura deux fonction de transitions δ_0 et δ_1 et, à chaque étape, on applique δ_0 (ou δ_1) avec probabilité $1/2$. De plus, tous ces choix sont indépendants.

Remarque 1. Les machines de Turing déterministes sont des cas particuliers car il suffit de poser $\delta_0 := \delta =: \delta_1$, en notant δ la fonction de transition de la machine de Turing déterministe.

Définition 1. La *probabilité d'accepter* $w \in \Sigma^\star$ est

$$\sum_{\substack{b \text{ chemin de calcul} \\ \text{acceptant sur l'entrée } w}} \Pr[b],$$

où, en notant k la longueur du chemin, [1]

$$\Pr[b] := \frac{1}{2^k}.$$

Définition 2. Pour $0 \leq \varepsilon < 1/2$, une machine probabiliste *reconnait* un langage A avec probabilité d'erreur ε si

1. Dans ce cas, c'est le nombre de tirages au sort.

1. si $w \in A$ alors $\Pr[w \text{ est accepté}] \geq 1 - \varepsilon$;
2. si $w \notin A$ alors $\Pr[w \text{ est rejeté}] \geq 1 - \varepsilon$.

On considère ensuite les machines probabilistes qui fonctionnent en temps polynomial en $|w|$. La machine s'arrêtera donc en un temps polynomial *peu importe* les choix probabilistes.

Définition 3. On définit la classe BPP [2] comme la classe des langages qui peuvent être reconnus par des machines probabilistes qui fonctionnent en temps polynomial avec probabilité d'erreur $1/3$.

De manière analogue à la classe NP, on a une caractérisation par certificats pour BPP.

Proposition 1. Un langage A est dans BPP si, et seulement si, il existe un polynôme p et un problème auxiliaire $B \in \mathsf{P}$ tel que

1. si $x \in A$ alors $\Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y\rangle \in B] \geq \frac{2}{3}$;
2. si $x \notin A$ alors $\Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y\rangle \notin B] \geq \frac{2}{3}$.

Le choix de $y \in \{0,1\}^{p(|x|)}$ se fait avec la distribution uniforme. □

Le choix de $\varepsilon = 1/3$ dans la définition de BPP peut sembler arbitraire mais, mais ça n'a pas d'importance comme le montre le lemme ci-dessous.

Lemme 1 (d'amplification). Soit $0 \leq \varepsilon < 1/2$. Supposons qu'un langage $A \subseteq \Sigma^\star$ est reconnu avec probabilité d'erreur ε par une machine de Turing probabiliste polynomiale M_1. Pour tout polynôme $p(n)$, alors A peut être reconnu par une machine probabiliste polynomiale M_2 avec probabilité d'erreur $2^{-p(n)}$ sur toute entrée de taille n.

2. Pour *Bounded Propability Polynomial*

Preuve. L'algorithme de M_2 est de la forme suivante.

```
1 : Effectuer 2k calculs de M_1 sur l'entrée x ∈ Σ^n
2 : si M_1 accepte pour une majorité (stricte) de calculs alors
3 :     Accepter
4 : sinon
5 :     Rejeter
```

Il nous reste à déterminer la valeur de k telle que

$$\Pr[M_2 \text{ fasse une erreur sur l'entrée } x] \leq \frac{1}{2^{p(n)}}.$$

Pour x donné, soit $S \in \{0,1\}^{2k}$ la suite des résultats obtenus à la ligne 1. Notons c le nombre de résultats corrects et w le nombre de résultats incorrects (on a donc $c + w = 2k$). On dit que S est « *mauvaise* » si $c \leq w$ (autrement dit $w \geq k$).

On a

$$\Pr[M_2 \text{ fasse une erreur sur l'entrée } x] \leq \sum_{S \text{ mauvaise}} \Pr[S].$$

On majore (grossièrement) le nombre de mauvaises suites S par le nombre total de suites, *i.e.* 2^{2k}. En fixant une mauvaise suite $S \in \{0,1\}^{2k}$, on a

$$\Pr[S] = \varepsilon(x)^w \cdot (1-\varepsilon)^c = \varepsilon(x)^k \cdot (1-\varepsilon)^k \cdot \left(\frac{\varepsilon(x)}{1-\varepsilon(x)}\right)^{w-k},$$

en notant $\varepsilon(x)$ la probabilité que M_1 fasse une erreur sur l'entrée x (on a donc $\varepsilon(x) \leq \varepsilon$). Comme $w \geq k$ (car S supposée mauvaise), on a

$$\left(\frac{\varepsilon(x)}{1-\varepsilon(x)}\right)^{w-k} < 1.$$

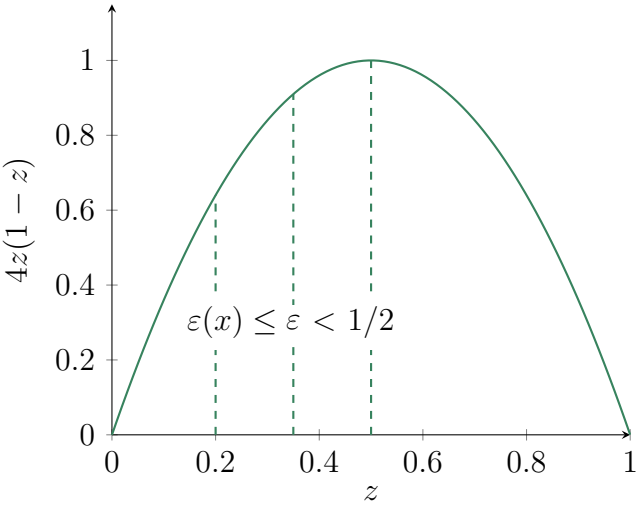

Figure 1 | *Graphe de* $z \mapsto 4z(1-z)$

On en déduit $\Pr[S] \leq \big(\varepsilon(x)(1-\varepsilon(x))\big)^k$, d'où

$$\begin{aligned}\Pr[M_2 \text{ fasse une erreur sur l'entrée } x] &\leq \big(4\varepsilon(x)(1-\varepsilon(x))\big)^k \\ &\leq \big(4\varepsilon(1-\varepsilon)\big)^k.\end{aligned}$$

Il suffit ainsi d'avoir $[4\varepsilon(1-\varepsilon)]^k \leq 2^{-p(n)}$, autrement dit,

$$k \geq \frac{p(n)}{-\log_2[4\varepsilon(1-\varepsilon)]}.$$

□

Proposition 2. On a BPP ⊆ PSPACE.

Preuve. On énumère tous les chemins de calcul possible, et on accepte si une majorité de ces chemins accepte. □

Théorème 1. On a BPP $\subseteq$ P/poly.

Preuve. Soit $A \in$ BPP et $B \in$ P tel que

1. si $x \in A$ alors $\Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y\rangle \in B] \geq 1 - \varepsilon(|x|)$
2. si $x \notin A$ alors $\Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y\rangle \notin B] \geq 1 - \varepsilon(|x|)$

où on va choisir $\varepsilon(|x|)$ exponentiellement petit (par lemme d'amplification).

Pour chaque n, on va fixer une valeur particulière $y_n \in \{0,1\}^{p(n)}$ pour la chaîne aléatoire y. Ce choix défini le conseil pour les entrées de taille n.

Montrons qu'il existe y_n qui donne une bonne réponse pour tout $x \in \{0,1\}^n$. On pose

$$f(x,y) := \begin{cases} 1 & \text{si } y \text{ conduit au bon résultat sur l'entrée } x \\ 0 & \text{sinon.} \end{cases}$$

On peut représenter ceci en un tableau à double entrées avec 2^n lignes et $2^{p(n)}$ colonnes. Notre but est donc de trouver une colonne qui ne contient que des « 1 ». On va supposer que la probabilité d'erreur est $\varepsilon(n) < 1/2^n$. Dans chaque ligne, on a moins (strict) de $2^{p(n)}/2^n$ zéros. Dans l'ensemble du tableau, on a moins (strict) de

$$\underbrace{2^n}_{\#\text{lignes}} \times \frac{2^{p(n)}}{2^n} = 2^{p(n)}$$

zéros. On a $2^{p(n)}$ colonnes donc il existe une colonne qui ne contient pas de zéros. □

Théorème 2. On a BPP $\subseteq \Sigma_2^{\mathrm{P}} \cap \Pi_2^{\mathrm{P}}$.

Preuve. Il suffit de montrer que BPP $\subseteq \Sigma_2^{\mathrm{P}}$ car coBPP = BPP. Soit $A \in$ BPP. Il existe donc p un polynôme et $B \in$ P tel que, pour tout $x \in \Sigma^n$ (on prend une probabilité d'erreur de $1/2^n$ par

le lemme d'amplification),

1. si $x \in A$ alors $\#\mathrm{Acc}(x) \geq \left(1 - \frac{1}{2^n}\right) \cdot 2^{p(n)}$;
2. si $x \notin A$ alors $\#\mathrm{Acc}(x) \leq 2^{p(n)}/2^n$,

en notant $\mathrm{Acc}(x) := \{y \in \{0,1\}^{p(|x|)} \mid \langle x, y\rangle \in B\}$.

Fixons $x \in \{0,1\}^n$. On doit donc distinguer (avec une formule du problème QBF-Σ_2^{P}) entre deux possibilités :

1. l'ensemble $\mathrm{Acc}(x)$ est « très grand » par rapport à $\{0,1\}^{p(n)}$;
2. l'ensemble $\mathrm{Acc}(x)$ est « très petit » par rapport à $\{0,1\}^{p(n)}$.

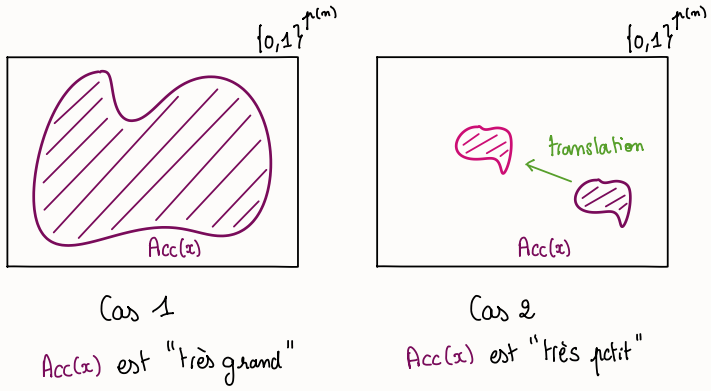

Figure 2 | *Les deux cas à distinguer*

L'idée est qu'on essaie de couvrir $\{0,1\}^{p(n)}$ par un petit nombre de translations de $\mathrm{Acc}(x)$, où la translation par $t \in \{0,1\}^{p(n)}$

$$\begin{aligned} \{0,1\}^{p(n)} &\longrightarrow \{0,1\}^{p(n)} \\ x &\longmapsto x \oplus t, \end{aligned}$$

où $\oplus$ est le XOR bit à bit. On pourra couvrir $\{0,1\}^{p(n)}$ uniquement dans le cas 1.

On va écrire

$$(*) \qquad x \in A \iff \exists t_1 \dots t_m \in \{0,1\}^{p(n)} \quad \forall z \in \{0,1\}^{p(n)} \quad \bigvee_{i=1}^{n} \langle x, z \oplus t_i \rangle \in B,$$

où les $t_1, \dots, t_m$ représentent les m vecteurs de translation.

Si cette équivalence est vraie avec m polynomial en n, on aura montré $A \in \Sigma_2^{\mathrm{P}}$. En effet, pour un certain z, on a

$$\langle x, z \oplus t_i \rangle \in B \iff z \oplus t_i \in \mathrm{Acc}(x) \iff z \in \mathrm{Acc}(x) \oplus t_i,$$

où $\mathrm{Acc}(x) \oplus t_i$ est le translaté de $\mathrm{Acc}(x)$. La formule $(*)$ dit donc que les m translations de $\mathrm{Acc}(x)$ recouvrent $\{0,1\}^{p(n)}$.

Lemme 2. Soit $E \subseteq \{0,1\}^k$. S'il existe $t_1, \dots, t_m \in \{0,1\}^k$ tels que

$$\bigcup_{i=1}^{m} (E \oplus t_i) = \{0,1\}^k,$$

alors $\#E \geq 2^k/m$.

Preuve. On a

$$\#\Big(\bigcup_{i=1}^{m} (E \oplus t_i) \Big) \leq \sum_{i=1}^{m} \#(E \oplus t_i) = m\, \#E.$$

□

Lemme 3. Si $2^k(1 - \#E/2^k)^m < 1$ alors il existe $t_1, \dots, t_m$ tels que $\bigcup_{i=1}^{m} (E \oplus t_i) = \{0,1\}^k$.

Remarque 2. On a

$$2^k\left(1-\frac{\#E}{2^k}\right)^m < 1 \quad \Longleftrightarrow \quad k\ln 2 + m\overbrace{\ln\left(1-\frac{\#E}{2^k}\right)}^{<-\frac{\#E}{2^k}} < 0,$$

donc il suffit d'avoir $k\ln 2 - m\#E/2^k < 0$, autrement dit

$$m > \frac{k\ln 2}{\#E/2^k}.$$

Preuve. On prend $t_1, \dots, t_m$ tirés au hasard suivant la distribution uniforme sur $\{0,1\}^k$ (choix indépendants et identiquement distribués, « iid »). Soit

$$\rho := \Pr\left[\bigcup_{i=1}^{m}(E \oplus t_i) \neq \{0,1\}^k\right]$$

la probabilité d'erreur. Si $\rho < 1$ alors on a le résultat. On remarque que $\rho \leq 2^k(1-\#E/2^k)^m$. Puis, on a

$$\begin{aligned}
\rho &= \Pr\left[\bigcup_{z\in\{0,1\}^k}\left(z \notin \bigcup_{i=1}^{m}(E\oplus t_i)\right)\right] \\
&\leq \sum_{z\in\{0,1\}^k}\Pr\left[z \notin \bigcup_{i=1}^{m}(E\oplus t_i)\right] \\
&= \sum_{z\in\{0,1\}^k}\Pr\left[\bigcap_{i=1}^{m}(z \notin E\oplus t_i)\right] \\
&= \sum_{z\in\{0,1\}^k}\prod_{i=1}^{m}\underbrace{\Pr[t_i \notin E\oplus z]}_{1-\#E/2^k},
\end{aligned}$$

et on conclut directement. □

On peut donc appliquer les deux lemmes avec $E = \mathrm{Acc}(x)$, $k = p(n)$ et $m = 2k$. On a que

- si $x \in A$ alors, par hypothèse $\#E/2^p \geq 1 - 1/2^n \geq 1/2$ (pour $n \geq 1$), par le second lemme, on a

$$m = 2k > 2k \ln 2 \geq \frac{k \ln 2}{\#E/2^k} ;$$

- si $x \notin A$ alors, par hypothèse $\#E/2^k \leq 1/2^n$, or pour tout couvrir, on devrait avoir

$$2p(n) \geq \frac{2^{p(n)}}{\#E} \geq 2^n,$$

ce qui est absurde.

□