IP = PSpace

Nathanaël François

19 décembre 2008

Résumé

Notre but est ici de s'intéresser à une généralisation de la classe NP à des preuves probabilistes. Lorsqu'on a un problème dans NP, on peut se donner une machine de Turing déterministe qui prend en entrée une instance du problème et un certificat de taille polynomiale, et vérifie qu'il s'agit effectivement d'une solution du problème. Ceci caractérise en fait exactement les problèmes NP, et on cherche à résoudre des problèmes plus complexes de manière similaire.

Pour cela, on adopte une condition moins restrictive : la machine de Turing est désormais randomisée, et on demande que la probabilité d'accepter une entrée soit sensiblement plus élevée si l'entrée est effectivement une instance du problème que si elle ne l'est pas. Le vérificateur peut pour cela poser des questions à un prouveur de capacité de calcul infinie, mais dont il n'est pas sûr a priori de l'honnêteté. Ceci définit la classe de complexité IP.

On va montrer que les problèmes que l'on peut résoudre de cette manière en temps polynomial sont en fait exactement ceux de la classe PSpace. On verra également, comme corollaire du résultat, que on peut se montrer plus restrictif dans la définition de IP, notamment imposer que toute instance du problème soit à coup sûr acceptée (mais on ne peut pas empêcher le vérificateur d'accepter des entrées qui ne sont pas des instances avec une petite probabilité).

1 Preuves interactives probabilistes

1.1 Définitions

Dans toute la suite, les machines que nous considérerons auront pour alphabet d'entrée Σ et pour alphabet de bande Γ. Nous allons considérer ici une généralisation de la résolution de problèmes par des machines de Turing. Pour cela, on définit un nouveau concept :

Définition 1 *Une machine de Turing randomisée est une machine de Turing déterministe disposant, en plus d'une bande d'entrée, d'une éventuelle bande de sortie et des bandes de travail, d'une bande d'aléa, également en lecture seule, contenant un mot de longueur infinie. La tête de lecture de la bande d'aléa est initialement en début de bande, et se déplace vers la droite à chaque transition.*

Comme la tête de lecture de la bande d'aléa n'est jamais située deux fois sur la même position, les bits "aléatoires" utilisés sont donc a priori indépendants les uns des autres.

On définit maintenant la classe IP des problèmes pouvant être vérifiés en temps polynômial de manière probabiliste. Comme ces preuves peuvent nécéssiter une interaction, on ne prend en compte dans le temps de calcul que celui de la vérification.

Définition 2 *On se donne $f : \mathbb{N} \mapsto \mathbb{N}$. On définit un vérificateur probabiliste comme une machine de Turing probabiliste $\mathcal{V}$ prenant en argument $< x, a_1, \ldots, a_{2r} >$, qui s'arrête en temps polynomial en la taille de x, renvoyant $< x, a_1, \ldots, a_{2r}, a_{2r+1} >$ si $r < f(|x|)$, et qui accepte ou refuse sinon. On définit un prouveur P, comme une machine de capacité de calcul supposée infinie, prenant en entrée $< x, a_1, \ldots, a_{2r+1} >$, renoyant $< x, a_1, \ldots, a_{2r+1}, a_{2r+2} >$ telle que a_{2r+2} soit de taille polynomial en $|x|$.*

Dans ces conditions, on définit $out_{\mathcal{V}}\langle P, \mathcal{V}\rangle(x)$ comme le résultat des $f(|x|)$ itérations du protocole défini par P et $\mathcal{V}$ sur l'entrée x.

Un langage $L \subset \Sigma^*$ est dans $IP[f]$ si il existe un vérificateur probabiliste tel que :
- *Si $x \in L$, il existe un prouveur P (prouveur honnête) tel que $out_{\mathcal{V}}\langle P, \mathcal{V}\rangle(x) = 1$ avec une probabilité supérieure à $\frac{2}{3}$. C'est l'hypothèse de complétude.*
- *Si $x \notin L$, pour tout prouveur P (prouveur éventuellement malhonnête), $out_{\mathcal{V}}\langle P, \mathcal{V}\rangle(x) = 0$ avec une probabilité supérieure à $\frac{2}{3}$. C'est l'hypothèse de sûreté.*

On pose $IP = \bigcup_{k\in\mathbb{N}} IP[n^k]$.

On voit que, quitte à répéter le processus plusieurs fois, la borne $\frac{2}{3}$ peut être arbitrairement améliorée. En effet la borne de Chernoff affirme que si Y est une variable aléatoire de loi $\mathcal{B}(p, N)$, $Pr[Y \geq (1+\theta)pN] \leq e^{-\frac{\theta^2}{3}pN}$ pour tout $\theta \in [0, 1[$.

En réalité, on peut faire bien mieux, comme on va le montrer plus tard. En effet, la classe des problèmes définis ainsi est exactement IP :

Un langage $L \subset \Sigma^*$ est dans IP$'$ si pour tout $l \geq 0$ il existe un vérificateur probabiliste tel que :
- Si $x \in L$, il existe un prouveur P (prouveur honnête) tel que l'on ait toujours $out_{\mathcal{V}}\langle P, \mathcal{V}\rangle(x) = 1$. La complétude est donc absolue.
- Si $x \notin L$, pour tout prouveur P (prouveur éventuellement malhonnête), $out_{\mathcal{V}}\langle P, \mathcal{V}\rangle(x) = 0$ avec une probabilité supérieure à $1 - 2^{-n^l}$. La sûreté peut donc être majorée par une exponentielle en un polynôme en n arbitraire.

On ne peut pas a priori faire mieux : si la sûreté était absolue, on retrouverait les problèmes de la classe NP.

1.2 Exemple de problème IP : le non-isomorphisme de graphe

Terminons cette partie par un exemple classique de problème de IP qui n'est pas a priori dans NP : le non-isomorphisme de graphe (GNI).

Deux graphes $G_1 = (\{1, \ldots, n_1\}, E_1)$ et $G_2 = (\{1, \ldots, n_2\}, E_2)$ sont dits isomorphes si $n_1 = n_2 = n$ et s'il existe $\sigma \in \mathcal{S}_n$ telle que $\forall i, j \in \{1, \ldots, n\}, ((i, j) \in E_1 \Leftrightarrow (\sigma(i), \sigma(j)) \in E_2)$.

Montrons que $L = \{< G_1, G_2 > | G_1 \text{ et } G_2 \text{ sont isomorphes}\} \in \text{IP}$.

On suppose que les deux graphes passés en argument sont toujours de même taille (ce qui peut se vérifier en temps linéaire), car sinon ils sont trivialement non isomorphes. On considère le protocole suivant :

Le vérificateur $\mathcal{V}$ choisit aléatoirement un G_i parmi G_1 et G_2, $\sigma \in \mathcal{S}_n$, et renvoie G l'image de G_i par l'isomorphisme de graphe induit par σ. P doit renvoyer un graphe isomorphe à G parmi G_1 et G_2 (on peut également lui demander un certificat d'isomorphisme, mais ce n'est pas nécéssaire). Si P renvoie effectivement G_i, $\mathcal{V}$ accepte, sinon $\mathcal{V}$ refuse.

Ce protocole convient s'il est effectué deux fois avant d'accepter. En effet, si G_1 et G_2 ne sont pas isomorphes, un prouveur honnête n'a qu'à renvoyer le seul graphe effectivement isomorphe à G pour que $\mathcal{V}$ accepte avec la probabilité 1. En revanche, s'ils sont isomorphes, P n'a aucun moyen de savoir si $\mathcal{V}$ a utilisé G_1 ou G_2. Par conséquent $\mathcal{V}$ refuse avec la probabilité $\frac{1}{2}$, et $\frac{3}{4} > \frac{2}{3}$ si on répète le protocole.

2 IP $\subseteq$ PSpace

Proposition 1 *IP* $\subseteq$ PSPACE

On va d'abord montrer ce résultat pour un sous-classe de IP : les preuves Arthur-Merlin polynômiales (en réalité, il y a égalité comme on le montrera en corollaire du théorème).

AM[f] est un protocole fonctionnnant comme IP[f] à la différence près que le vérificateur (Arthur) révèle au prouveur tous les bits d'aléa qu'il a utilisés, plutôt que de les garder secrets et de ne poser que des questions.

Définition 3 *On définit un jeu entre Arthur et Merlin comme un triplet* $(M, \mathcal{A}, D)$*, où :*

- *M est une fonction, éventuellement non calculable, qui à un mot de Σ^* associe un mot de longueur polynômiale en la longueur de son plus grand préfixe sans #.*
- *$\mathcal{A}$ est une machine de Turing probabiliste (i.e. possèdant une bande dite d'aléa, en lecture seule, dont le contenu est indépendant de l'entrée), disposant d'une bande de sortie, et qui s'arrête en temps polynômial en la longueur du plus grand préfixe de l'entrée sans #.*
- *D est un langage sur Γ^* dans P.*

On se donne prot *un mot de* $\{\text{A}, \text{M}\}^*$*, on découpe la bande d'aléa en sous-mots r_j de longueur égale, polynômiale en la longueur n de l'entrée, et on effectue le protocole suivant :*

Input x
inp := x *// contient l'entrée fournie aux joueurs*

$j := 0$ // *nombre de fois où Arthur a joué*
For $i = 1$ ***to*** $|\text{prot}|$ ***do***
If $\text{prot}_i = \text{A}$
Then $j : j + 1$; $\text{inp} := \text{inp}\#r_j\#\mathcal{A}(\text{inp}, r_j)$; // *Arthur pose une question aléatoire.*
Else $\text{inp} := \text{inp}\#M(\text{inp})$ //*Merlin répond.*
Accepter si et seulement si $\text{inp} \in D$

Pour toute fonction $f : \mathbb{N} \mapsto \mathbb{N}$*, on note AM[f] la classe des langages L tels que :*

- *Si* $x \in L$ *alors le protocole associé à* $(\text{AM})^{f(n)}$ *accepte avec une probabilité supérieure à* $\frac{2}{3}$
- *Si* $x \notin L$ *alors le protocole associé à* $(\text{AM})^{f(n)}$ *accepte avec une probabilité inférieure à* $\frac{1}{3}$

On note $\text{ABPP} = \bigcup_{k \in \mathbb{N}} \text{AM}[n^k]$.

On voit qu'il existe une inclusion triviale ABPP $\subseteq$ IP. En effet, rien n'empêche a priori le vérificateur de révéler ses bits d'aléa dans un protocole IP.

L'inclusion réciproque n'est pas immédiate (et n'est pas vraie dans le cas général : on a seulement $\text{IP}[f] \subset \text{ABPP}[f + 2]$). On voit par exemple que l'algorithme donné pour montrer que GNI est dans IP ne fonctionne pas si les bits d'aléa d'Arthur sont connus par Merlin, car alors Merlin peut systématiquement renvoyer le graphe choisi par Arthur.

Montrons donc ABPP $\subseteq$ PSpace, et on s'inspirera de la preuve pour étendre le résultat à IP $\subseteq$ PSpace.

Comme PSpace = NPSpace (théorème de Savitch), on peut se contenter de montrer que tout protocole de ABPP peut être simulé par une machine de Turing non déterministe en espace polynomial. On suppose que L est décidé par un protocole *prot* à $2f(n) + 1$ tours, que $q(n)$ bits d'aléa sont utilisés au plus à chaque tour, et que les réponses de Merlin sont au plus de taille $p(n)$.

f, q et p sont majorés par des polynomes, donc quitte à rajouter des tours inutiles, elles sont calculables en espace polynomial.

On définit donc la fonction *Simul* ainsi :

let rec $Simul(inp, j, i)$ //i nombre de tours joués, j nombre de tours où Arthur a joué
if $i = |prot|$ //le protocole est fini
then if $inp \in \mathcal{D}$
then $cnt := cnt + 1$; //Sur cet aléa, le protocole accepte
else ;
else if $prot_i = \text{A}$ //C'est à Arthur de jouer
then for $r \in \{0,1\}^{q(n)}$ **do** $Simul(inp\#r\#\mathcal{A}(inp, r), j + 1, i + 1)$;
else { **guess** $y \in \{0,1\}^{q(n)}$; $Simul(inp\#y, j, i+1)$ } ; //On devine la réponse de Merlin qui maximise *cnt*.

On calcule ensuite $Simul(x, 0, 0)$, et on compare *cnt* (initialisé à 0) à $\frac{1}{2}2^{f(n)q(n)}$.

On a tout fait dans la définition pour s'assurer que Simul était bien en espace polynômial. Donc ABPP $\subseteq$ PSpace.

Adaptons maintenant à IP $\subseteq$ PSpace. La subtilité est que l'on ne peut pas faire une fonction récursive *Simul* dont *secret* (les bits d'aléas) serait un argument, car alors y pourrait dépendre de *secret*, ce que l'on interdit.

Il faut donc plutôt faire une fonction *Simul* prenant en argument uniquement *public* et qui effectue la récurrence sur toutes les questions plutôt sur les bits d'aléas. On appelle à l'intérieur de cette fonction une fonction *Compte* qui renvoie le nombre de valeurs de *secret* telle que la valeur de *public* obtenue fasse accepter le protocole (pour l'incrémentation de *cnt*). Ces deux fonctions sont encore en espace polynômial. Donc on a prouvé l'inclusion IP $\subseteq$ PSpace.

3 IP = PSpace

Théorème 1 (Shamir) *ABPP = IP =* PSpace

3.1 Preuve du théorème : QSAT est dans ABPP

Il suffit, pour prouver le théorème, de montrer que PSpace $\subseteq$ ABPP, car on a déjà les deux autres inclusions. Par ailleurs, comme Arthur est autorisé à faire n'importe quel calcul en temps polynômial, clairement, si A $\leq_{\mathrm{P}}$ B et B $\in$ ABPP, alors B $\in$ ABPP. En particulier, il suffit de montrer QSAT $\in$ ABPP, car QSAT est PSpace-complet. On va de plus montrer que QSAT est encore dans ABPP si on prend une définition plus restrictive de cette dernière classe pour la complètude et la sûreté.

Le lemme de Schwartz-Zippel, utilisé dans la preuve, sera démontré plus loin.

Soit F une formule booléenne quantifiée close. On peut la supposer en forme normale prénexe.

$F = Q_1 x_1 \cdot \ldots \cdot Q_m x_m G(x_1, \ldots, x_m)$, avec $Q_i \in \{\forall \exists\}$ et G sans quantificateurs. On peut supposer G en forme normale conjonctive.

L'idée est ici d'interprèter F comme un polynôme à m variables sur un corps fini. $x \wedge y$ se traduit par xy, $\neg x$ par $1 - x$ et $x \vee y$ par $x + y - xy$. Cette transformation est également utile dans d'autres preuves reliant les classes de complexité randomisées aux classes de complexité classiques.

On traduit également les quantificateurs en termes de polynômes par $\forall x \cdot P = P[x := 0] \times P[x := 1]$ et $\exists x \cdot P = P[x := 0] + P[x := 1] - P[x := 0] \times P[x := 1]$. De plus, on définit $Rx \cdot P = P \mod (x^2 - x)$. $Rx \cdot P$ prend les mêmes valeurs que P sur les booléens, mais est de degré au plus 1 en x, ce qui permet d'éviter que le degré du polynôme n'explose avec les quantificateurs. On pose $\hat{F}$ et $\hat{G}$ les polynômes ainsi obtenus à partir de F et G.

Il s'agit ensuite de demander à Merlin de construire une suite de polynômes à partir de G jusqu'à F. On impose une certaine cohérence entre les polynômes qui permet de savoir si Merlin triche avec une probabilité élevée. Voici comment on procède :

Comme on réduit le degré, on a :
$\hat{F} = Q_1 x_1 \cdot Rx_1 \cdot Q_2 x_2 \cdot Rx_1 \cdot Rx_2 \cdot \ldots \cdot Q_m x_m \cdot Rx_1 \ldots Rx_m \cdot \hat{G}(x_1 \ldots x_m)$

On pose, pour $0 \leq j < i \leq m$:
$P_{(i,j)}(x_1, \ldots, x_i) = Rx_{j+1} \cdot \ldots \cdot Rx_i \cdot$
$Q_i x_i \cdot Rx_1 \cdot \ldots \cdot Rx_i \cdot$
$\ldots$
$Q_m x_m \cdot Rx_m \ldots Rx_1 \cdot \hat{G}(x_1 \ldots x_m)$

On a donc $P_{(0,0)} = \hat{F}$, $P_{(m+1,0)} = \hat{G}$. Le successeur de $P_{(i,j)}$ étant naturellement le polynôme obtenu en retirant l'opérateur le plus à gauche, on pose $s(i,j) = (i, j+1)$ si $j+1 < i$, et $s(i,j) = (i+1, 0)$ sinon.

On va stocker dans l'entrée tous les polynômes $P_{(i,j)}$, que l'on demande à Merlin de calculer en partant de $\hat{F}$. On a le droit de faire cette opération, car la taille de la nouvelle entrée reste polynômiale en la taille de la formule F. En effet, le degré maximal des polynômes que l'on empile est borné par $d = \max(2 * m, \deg(\hat{G}))$, on en empile $q = O(m^2)$, et $\deg(\hat{G})$ est borné par le nombre de connecteurs logiques dans $\hat{G}$. On stockera également les bits aléatoires d'Arthur, qui seront forcément de taille polynômiale car Arthur calcule en temps polynômial.

Voici comment on procède :

Soit $l \geq 0$, n la taille de l'entrée F. Arthur demande à Merlin un nombre premier p entre $2^{n^l} d(q+1)$ et $2^{n^l+1} d(q+1)$, ce qui est possible par le postulat de Bertrand, avec un certificat de primalité de taille polynômiale. p est bien de taille polynômiale en n.

A chaque tour, on effectue le protocole suivant :

1. Merlin fournit un polynôme $\tilde{P}_{(i,j)}(x_j)$, qu'il prétend égal à $P_{(i,j)}(v_1, \ldots, v_{j-1}, x_j)$, où $v_1, \ldots, v_{j-1}$ sont des valeurs aléatoires de $\mathbb{F}_p$ précédemment tirées au hasardpar Arthur.
2. Arthur vérfie que $\tilde{P}_{(i,j)}(x_j)$ est de degré d au plus (en réalité c'est bien moins). Cette étape permet d'éviter de perdre un temps non polynômial si Merlin renvoie n'importe quoi.
3. – Si $(i,j) = (0,0)$, Arthur vérifie que $\tilde{P}_{(i,j)}$ est la constante 1.
   – Si $j = 0$ et $i > 0$, Arthur $Qx_j \cdot \tilde{P}_{(i,j)}(x_j) = w$, où w était l'objectif fixé au tour précédent.
   – Si $j > 0$, Arthur $Rx_j \cdot \tilde{P}_{(i,j)}(v_j) = w$, où v_j est un élément de $\mathbb{F}_p$ fixé au tour précédent, w l'objectif.
4. Arthur tire une (nouvelle) valeur aléatoire v_j pour x_j, et calcule $w = \tilde{P}_{(i,j)}(v_j)$ le nouvel objectif. On passe au tour suivant (ou on sort de la boucle si (i,j) $= (m+1, 0)$)

A la fin du jeu, Arthur vérifie $\hat{G}(v_1, \ldots, v_m) = \tilde{P}_{(m+1,0)}(v_1, \ldots, v_m)$ et accepte si c'est le cas. Le calcul de $\hat{G}$ est polynômial car il est sans quantificateurs.

Il nous reste à montrer que ce protocole satisfait bien les conditions. On voit que si F est effectivement satisfaite, si Merlin est honnête (i.e. il fournit à chaque fois le bon polynôme), Arthur accepte à tous les coups.

On suppose désormais que F n'est pas satisfaite. Merlin doit tricher pour que Arthur accepte, car sinon il s'aperçoit que $P_{(0,0)} = 0$.

Si Merlin triche sur $\tilde{P}_{(m+1,0)}$, qu'il ne prend pas égal à $\hat{G}$ mais donne des polynômes cohérents entre eux, il doit faire ce choix avant de connaître les bits aléatoires d'Arthur. Le lemme de Schwartz-Zippel affirme alors que comme $\hat{G} - \tilde{P}_{(m+1,0)} \neq 0$, et $\deg(\hat{G} - \tilde{P}_{(m+1,0)}) < p$, la probabilité que $\hat{G}(v_1, \ldots, v_m) = \tilde{P}_{(m+1,0)}(v_1, \ldots, v_m)$ est inférieure à $\frac{d}{p}$.

Sinon, Merlin utilise le bon $\tilde{P}_{(m+1,0)}$, mais attend une opportunité pour tricher au cours du jeu. En effet, lorsque $j > 0$, comme il connaît v_j il peut aisément fournir un polynôme faux qui passera le test d'Arthur, mais si ce polynôme n'est pas cohérent avec $\hat{G}$, Merlin n'a rien gagné et doit tricher à nouveau.

A chaque étape, la probabilité que Merlin ait une opportunité de tricher est, toujours d'après le lemme de Schwartz-Zippel, $\frac{d}{p}$. Merlin réussit donc à tricher sans qu'Arthur s'en aperçoive avec une probabilité au plus $(q+1)\frac{d}{p} \leq 2^{-n^l}$.

Il en résulte QSAT $\in$ ABPP, donc ABPP = IP = PSPACE, et que les deux définitions de IP sont équivalentes. □

3.2 Le lemme de Schwartz-Zippel

Lemme 1 (Schwartz-Zippel) *Soit p un nombre premier, $m \geq 1$, P un polynôme non nul de $\mathbb{F}_p[X_1, \ldots, X_m]$. On suppose que $\deg(P) < p$. Alors si $v_1, \ldots, v_m$ sont des variables aléatoires indépendantes de loi uniforme dans $\mathbb{F}_p$, la probabilité que $P(v_1, \ldots, v_m) = 0$ est au plus $\frac{d}{p}$.*

Il suffit pour cela de montrer que P a au plus dp^{m_1} racines. On procède par récurrence sur $m \geq 1$.

Si $m = 1$, c'est trivial.

On suppose donc $m \geq 2$. On peut écrire P comme un polynôme en X_m à coefficients dans $\mathbb{F}_p[X_1, \ldots, X_{m-1}]$. Soit d_m son degré et $Q_m(X_1, \ldots, X_{m-1})$ son coefficient dominant. Le degré de Q_m est au plus $d - d_m$. Si $P(v_1, \ldots, v_m) = 0$, soit $Q_m(v_1, \ldots, v_{m-1}) = 0$, ce qui arrive pour au plus $(d - d_m)p^{m-2}$ valeurs de $(v_1, \ldots, v_{m-1})$, donc au plus $(d - d_m)p^{m-1}$ valeurs de $(v_1, \ldots, v_m)$, soit $P(v_1, \ldots, v_{m-1}, X_m)$ s'annule en v_m, ce qui arrive pour au plus d_m valeurs de v_m, donc au plus $d_m p^{m-1}$ valeurs de $(v_1, \ldots, v_m)$. Donc P a au plus dp^{m_1}. □

4 Références

[1] Jean-Goubault-Larrec, *Classes de complexité randomisées*

[2] Sanjev Arora, Boaz Barak, *Computational complexity : A modern approach* (p.147-162)

[3] Rajeev Motwani, Prabhakar Raghavan, *Randomized Algorithms* (p.172-180)