Mémo complexité

P. Pansu

31 mars 2009

1 P, NP, BPP

1.1 Conventions

Instances = éléments de $\{0,1\}^*$
Algorithme = machine de Turing déterministe
Algorithme polynômial = machine M qui retourne $M(x)$ en temps polynômial en $|x|$
Problème de décision= langage = sous-ensemble de $\{0,1\}^*$ = fonction booléenne sur $\{0,1\}^*$

1.2 P et NP

Définition 1 *La classe* **P** *: $L \in$ **P** s'il existe un algorithme polynômial M tel que, pour toute instance x,*

$$M(x) = L(x).$$

Définition 2 *La classe* **NP** *: $L \in$ **NP** s'il existe un algorithme polynômial M (le* vérificateur*) et un exposant d tels que, pour toute instance x,*

$$x \in L \Leftrightarrow \exists u,\ |u| \leq |x|^d,\ t.q.\ M(x,u) = 1.$$

On appelle u un certificat *pour x.*

Exemple 3 *Une instance de* SAT*, c'est une famille de clauses booléennes sous-forme normale conjonctive (un ET de OU en les variables ou leurs négations). Un certificat, pour une instance de* SAT*, c'est un choix de valeurs pour les variables qui satisfait toutes les clauses données.* 3SAT *est le même problème, restreint aux clauses dans lesquelles n'interviennent que 3 variables (et leurs négations) à la fois (i.e. chaque clause est de la forme $\bar{x}_i \vee x_j \vee \bar{x}_k$, par exemple).*

1.3 Réduction au sens de Karp

Définition 4 *Une* réduction polynômiale *de L à L', c'est une application f calculable en temps polynômial telle que*

$$x \in L \Leftrightarrow f(x) \in L'.$$

L est **NP***-dur si, pour tout langage $L'' \in$ **NP**, il existe une réduction polynômiale de L'' à L. L est* **NP***-complet si L est* **NP** *et* **NP***-dur.*

Exemple 5 SAT *se réduit polynômialement à* 3SAT*.*

Théorème 1 (Cook-Levin 1971)*.* 3SAT *est* **NP***-complet.*

Exemple 6 *Il existe des réductions polynômiales de* 3SAT *à* INDSET*, puis à* VERTEXCOVER*, puis à* MAX-CUT*.*

1.4 Algorithmes probabilistes

Définition 7 *La classe* **BPP** *: $L \in$ **BPP** s'il existe un algorithme polynômial M et un exposant d tels que, pour toute instance x,*

$$Pr_{|r|=|x|^d}(M(x,r) = L(x)) \geq \frac{2}{3}.$$

Remarque 8 *Pour tout $e > 0$, on peut remplacer $\frac{2}{3}$ par $\frac{1}{2} + |x|^{-e}$ ou par $1 - 2^{-|x|^e}$ sans que cela change la classe obtenue.*

En effet, il suffit de faire tourner la machine plein de fois.

Exemple 9 FindKthElement. *Il s'agit de trouver le k-ème par ordre croissant dans une série de n valeurs numériques. Voici un algorithme probabiliste qui le résoud en temps linéaire (alors que le tri donnerait $O(n \log n)$).*

- *En tirer un terme de la série au hasard, soit x, compter le nombre m de termes $\leq x$.*
- *Si $m > k$, relancer l'algorithme sur la liste écrémée.*
- *Sinon, relancer sur le complémentaire (en changeant k en $k - m$).*

2 Difficulté d'approximation

2.1 Problèmes d'optimisation

Exemple 10 *Une contrainte booléenne, c'est une fonction booléenne arbitraire de variables booléennes. Une instance de qCSP$_W$ est un système de contraintes booléennes écrites dans un alphabet à W lettres, chacune faisant intervenir q variables. Une instance de* MAX-qCSP$_W$*, c'est la même chose. Une solution est une affectation quelconque des variables. La valeur d'une affectation est la proportion de clauses qui sont satisfaites. Le problème consiste, étant donnée une instance ϕ, à déterminer $val(\phi) = \max\{val(u) \mid u \text{ affectation}\}$.*

Exemple 11 *Une instance de* MAX-3SAT *est la même chose qu'une instance de* 3SAT*, i.e. une famille de clauses booléennes sous-forme normale conjonctive. Une solution est une affectation quelconque des variables. La valeur d'une affectation est la proportion de clauses qui sont satisfaites. Le problème consiste, étant donnée une instance ϕ, à déterminer $val(\phi) = \max\{val(u) \mid u \text{ affectation}\}$.*

Définition 12 3LIN *est un sous-problème de* 3SAT *: celui où les clauses booléennes sont de la forme $x_i + x_j + x_k = y$ modulo 2.* MAX-3LIN *est la version optimisation de* 3LIN*.*

Sur les 8 clauses booléennes en 3 variables, 2 sont linéaires.

Remarque 13 3LIN$\in$ **P**.

En effet, l'algèbre linéaire permet de résoudre les systèmes d'équations linéaires en temps polynômial. En revanche, si $c < 1$, le problème, noté 3LIN$_c$, de savoir si, pour un système linéaire, une fraction au moins c des équations ont une solution commune est difficile (voir ci-dessous).

Exemple 14 *Une instance de* MAX-CUT *est un graphe fini G. Sa taille est le nombre de sommets. Une solution est une coupe, i.e. un sous-ensemble S de l'ensemble des sommets. La valeur de S est le nombre d'arêtes qui relient un sommet de S à un sommet du complémentaire. Le problème consiste, étant donné une instance G, à déterminer $val(G) = \max\{val(S) \mid S \text{ coupe}\}$.*

2.2 Résolution approchée

Définition 15 *Etant donné un problème d'optimisation, un algorithme polynômial d'*$\alpha(n)$*-approximation est un algorithme qui, en temps polynômial en la taille* n *de l'instance* x*, retourne une solution* u *dont la valeur est* $\geq \alpha(n)val(x)$*.*

Exemple 16 *Etant donné* $\epsilon > 0$*, voici un algorithme (probabiliste) polynômial de* $\frac{7}{8} - \epsilon$*-approximation de* MAX-3SAT *:*

1. *effectuer* k *tirages au hasard de valeurs des variables ;*
2. *retourner celui qui satisfait le plus de clauses.*

Il peut être dérandomisé, à l'aide de programmation semi-définie.

En effet, l'espérance de la valeur $val(u)$ d'une affectation u tirée au hasard vaut $\frac{7}{8}$, car chaque clause est satisfaite par 7 affectations d'un triplet de variables sur 8. La fraction de clauses satisfaites suit une loi binômiale $\mathcal{B}(\frac{7}{8}, k)$, donc la probabilité que, pour tous les tirages, le score soit $< \frac{7}{8} - \epsilon$, est inférieure à $\frac{1}{3}$ pour k assez grand.

Exemple 17 *Etant donné* $\epsilon > 0$*, voici un algorithme (probabiliste) polynômial de* $\frac{1}{2} - \epsilon$*-approximation de* MAX-3LIN *:*

1. *effectuer* k *tirages au hasard de valeurs des variables ;*
2. *retourner celui qui satisfait le plus de clauses.*

En effet, l'espérance de la valeur $val(u)$ d'une affectation u tirée au hasard vaut $\frac{1}{2}$, car chaque clause est satisfaite par 4 affectations d'un triplet de variables sur 8.

Théorème 2 (Goemans-Williamson 1995) MAX-CUT *possède un algorithme (probabiliste) polynômial de* ρ*-approximation, où* $\rho = \frac{2}{\pi \sin(\theta)} = 0.978567...$ *où* θ *est la solution de* $\theta = \tan(\frac{\theta}{2})$*.*

Il peut-être dérandomisé (Mahadjan-Ramesh 1999).

2.3 Rapport d'approximation

Définition 18 *Si* O *est un problème d'optimisation, et* $0 \leq s < c$*, on note* (s, c)-GAP-O *le problème suivant : étant donné une instance* x *de* O*, décider si* $val(x) \geq c$ *ou bien* $val(x) \leq s$*. Une* réduction *d'un problème de décision* L *à* (s, c)-GAP-O *est une application* f *calculable en temps polynômial* f *qui transforme toute instance de* L *en une instance de* O *telle que*

- $x \in L \Rightarrow val(f(x)) \geq c$ *;*
- $x \notin L \Rightarrow val(f(x)) \leq s$*.*

On peut donc parler de problèmes d'optimisation approchés **NP***-difficiles.*

c s'appelle la *complétude*, `completeness`, s la *sûreté*, `soundness`.

Théorème 3 (Théorème **PCP**, Arora-Safra 1998, [2]) *Il existe* $s < 1$ *tel que* $(s, 1)$-MAX-qCSP$_2$ *est* **NP***-difficile.*

Le Théorème **PCP** entraîne la difficulté d'approximation de nombreux problèmes d'optimisation, sans donner en général de facteur optimal (attention, les réductions ne préservent pas le rapport c/s). C'est néanmoins le cas pour MAX-INDSET ou MAX-VERTEXCOVER, par exemple. Pour MAX-3LIN et MAX-3SAT, on connaît le rapport optimal, mais il faut davantage d'arguments.

Théorème 4 (Håstad 2001) *Pour tout* $\epsilon > 0$*,* $(\frac{1}{2} + \epsilon, 1 - \epsilon)$-Gap-MAX-3LIN *est* **NP***-difficile.*

Corollaire 19 (Håstad 2001) *Pour tout* $\epsilon > 0$*,* $(\frac{7}{8} + \epsilon, 1 - \epsilon)$-Gap-MAX-3SAT *est* **NP***-difficile.*

Preuve : facile à déduire du Théorème 4.

Corollaire 20 *Pour tout* $\epsilon > 0$*,* $(\frac{16}{17} + \epsilon, 1 - \epsilon)$-Gap-MAX-CUT *est* **NP***-difficile.*

La valeur $\frac{16}{17}$ n'est pas optimale, c'est celle du théorème de Goemans-Williamson qui est conjecturée comme optimale.

3 Preuves localement vérifiables

3.1 Ne lire qu'une infime partie de la preuve

Certains problèmes (langages) possèdent un vérificateur probabiliste particulièrement économe : armé de $r(n)$ bits aléatoires, il lui suffit de consulter $q(n)$ bits du certificat pour décider avec une faible probabilité d'erreur. On adopte un vocabulaire mathématique : le certificat est vu comme une preuve de l'assertion $x \in L$. Lorsque $x \in L$, on exige l'*existence* d'une preuve acceptée à tous les coups (complétude). Mais lorsque $x \notin L$, toute preuve sera rejetée avec une probabilité significative.

Définition 21 *Un vérificateur $(r(n), q(n))$-probabilistiquement correct pour un problème L est un algorithme polynômial M qui a accès à $r(n)$ bits aléatoires, n'utilise que $q(n)$-bits de la preuve π (dont les positions dépendent de r) et qui possède les propriétés suivantes.*

- *Complétude. Si $x \in L$, il existe une preuve π telle que $Pr(M(x, r, \pi) = 1) = 1$.*
- *Sûreté. Si $x \notin L$, pour toute preuve π, $Pr(M(x, r, \pi) = 1) \leq \frac{1}{2}$.*

On dit que $L \in \mathbf{PCP}(r(n), q(n))$ s'il existe une constante C telle que L possède un vérificateur $(Cr(n), Cq(n))$-probabilistiquement correct.

Théorème 5 (Théorème **PCP**, version système de preuve) $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$.

L'équivalence entre difficulté d'approximation et preuves probabilistiquement vérifiables a été dégagée dans [4]. Voir aussi [3].

3.2 Lire seulement 3 bits d'une preuve

Dans le Théorème 5, quel que soit le problème étudié, le niveau de sûreté $\frac{1}{2}$ peut être remplacé par n'importe quel $s > 0$ en relançant le vérificateur sur la même preuve plusieurs fois. Mais alors, le nombre de requêtes q augmente. De même, on peut se limiter à 3 requêtes, mais alors le niveau de sûreté se rapproche de 1. D'où une amélioration possible.

Théorème 6 (Håstad 2001) *Pour tout $\epsilon > 0$, $\mathsf{3LIN}$ possède un vérificateur $(\log n, 3)$-probabilistiquement correct, avec un niveau de complétude $1 - \epsilon$ et un niveau de sûreté $\frac{1}{2} + \epsilon$.*

C'est de cette version du théorème **PCP** que découlent les résultats optimaux (Théorème 4 et Corollaire 19).

4 Démonstration du Théorème PCP

4.1 Schéma

On suit la preuve donnée par [1], qui repose sur le Lemme d'amplification d'Irit Dinur.

On part du problème **NP**-complet $q\mathsf{CSP}_2$. Si on a m-contraintes, si une instance $\phi \notin q\mathsf{CSP}_2$, alors $val(\phi) \leq 1 - \frac{1}{m}$. Par conséquent, $(1 - \frac{1}{m}, 1)$-$\mathsf{GAP\text{-}3CSP}_2$, équivalent à $\mathsf{3SAT}$, est aussi **NP**-complet. On construit une suite de vérificateurs dont la sûreté augmente jusqu'à atteindre une valeur indépendante de m.

Lemme 22 (Amplification). *Si on accepte d'augmenter la taille de l'alphabet, il existe une réduction g de $q\mathsf{CSP}_2$ vers $2\mathsf{CSP}_W$ qui diminue la valeur des instances. I.e., étant donné q, il existe W, g et $\epsilon_0 > 0$ tels que, si $\epsilon < \epsilon_0$,*

$$val(\phi) \leq 1 - \epsilon \Rightarrow val(g(\phi)) \leq 1 - 6\epsilon.$$

De plus, le nombre de contraintes m est multiplié par une constante, au pire.

Lemme 23 (Réduction de l'alphabet). *Si on accepte d'augmenter l'arité, on peut diminuer la taille de l'alphabet. I.e., il existe une réduction de* $2\mathsf{CSP}_W$ *vers* $q\mathsf{CSP}_2$ *qui augmente modérément la valeur des instances,*

$$val(\phi) \leq 1 - \epsilon \Rightarrow val(h(\phi)) \leq 1 - \epsilon/3.$$

De plus, le nombre de contraintes m *est multiplié par une constante, au pire.*

En appliquant les lemmes 22 et 23 $\log m$ fois, on obtient une réduction polynômiale de $q\mathsf{CSP}_2$ à $(1 - \epsilon_0, 1)$-GAP-$q\mathsf{CSP}_2$.

4.2 Amplification

4.3 Réduction de l'alphabet

4.4 Répétition parallèle

4.5 Démonstration du théorème PCP en 3 bits

4.6 Tester la linéarité en 3 bits

4.7 Reconnaître les juntes en 3 bits

5 Sujets d'exposés

1. Théorème 3 $\Leftrightarrow$ Théorème 5 : deux interprétations du Théorème **PCP**, d'après [1], section 11.3.
2. Théorème 4 $\Rightarrow$ Théorème 20 : réduction de MAX-3LIN à MAX-CUT, d'après [6].
3. Version bébé du théorème **PCP** : $\mathbf{NP} \subset \mathbf{PCP}(\text{poly}, 1)$, d'après [1], section 11.5.
4. Dérandomisation des algorithmes d'approximation triviaux pour MAX-3SAT et MAX-3SAT, d'après ? ? ? ? ?
5. Dérandomisation de l'algorithmes d'approximation de Goemans et Williamson pour MAX-CUT, d'après [7].

Références

[1] Arora, Sanjeev ; Barak, Boaz ; Complexity Theory : A Modern Approach. To be published by Cambridge University Press, around March 2009.

[2] Arora, Sanjeev ; Safra, Shmuel ; Probabilistic checking of proofs : a new characterization of NP. (English summary) J. ACM **45** (1998), no. 1, 70–122.

[3] Arora, Sanjeev ; Lund, Carsten ; Motwani, Rajeev ; Sudan, Madhu ; Szegedy, Mario ; Proof verification and the hardness of approximation problems. (English summary) J. ACM **45** (1998), no. 3, 501–555.

[4] Feige, Uriel ; Goldwasser, Shafi ; Lovász, Laszlo ; Safra, Shmuel ; Szegedy, Mario ; Interactive proofs and the hardness of approximating cliques. J. ACM **43** (1996), no. 2, 268–292.

[5] Håstad, Johan ; Some optimal inapproximability results. J. ACM **48** (2001), no. 4, 798–859.

[6] Trevisan, Luca ; Sorkin, G.B. ; Sudan, Madhu ; Williamson, D.P. ; Gadgets, Approximation, and Linear Programming. SIAM J. on Computing, **29(6)** (2000) 2074–2097.

[7] Engebretsen, Lars ; Indyk, Piotr ; O'Donnell, Ryan ; Derandomized dimensionality reduction with applications. 13th Symposium on Discrete Algorithms, 2002.