par Nicolas Schabanel

Vérifier une preuve
en n'en lisant que quelques lettres !

Le théorème PCP affirme qu'il est possible d'être convaincu de la validité d'une preuve par la seule lecture de trois *bits* ! En outre, la preuve écrite sous la forme qui convient n'est pas beaucoup plus longue que la preuve originale. Comment un tel miracle peut-il être possible ?

Au début des années 1990, une véritable révolution s'est opérée en informatique dans la théorie de la démonstration, réunissant différents champs de l'informatique qui s'étaient développés indépendamment durant la décennie précédente : les codes correcteurs d'erreurs, la cryptographie, l'auto-correction de programmes, les preuves interactives, la dérandomisation et, de façon assez inespérée, l'optimisation. Cette révolution est la découverte du théorème PCP en 1992 par Sanjeev Arora, Shmuel Safra, Carsten Lund, Rajeev Motwani, Madhu Sudan et Mario Szegedy. Le caractère révolutionnaire de ce théorème est qu'il démontre que l'on peut écrire une preuve de n'importe quel résultat mathématique de sorte que la lecture d'un nombre constant (une constante absolue !) de symboles, à des positions choisies au hasard dans la preuve, suffit pour être convaincu que le résultat est effectivement correct. La notion de « résultat mathématique » est à prendre au sens large d'une solution vérifiable classiquement en temps raisonnable (c'est-à-dire NP) à un problème algorithmique (comme trouver un chemin de longueur minimale passant par toutes les villes d'un pays). En outre, la preuve écrite sous cette forme n'est pas beaucoup plus longue que la preuve originale, mais infiniment plus rapide à vérifier et donc en un certain sens… infiniment plus fiable !

Un théorème vraiment surprenant !

Ce résultat historique, qui sera complètement détaillé dans les pages suivantes, est l'aboutissement d'une série de travaux portant sur différents domaines de l'informatique *a priori* assez éloignés. Tout d'abord, on trouve la théorie des codes correcteurs d'erreurs et de l'auto-correction de programme, avec les codes localement testables de Walsh–Hadamard, certes extrêmement longs à écrire mais dont on peut tester la correction puis extraire des valeurs correctes en n'en lisant qu'un nombre constant de *bits*. Les graphes expandeurs jouent ensuite un rôle important

car ils réunissent deux caractéristiques *a priori* contradictoires : la densité d'un arbre et la capacité de mélange d'un graphe complet. Apparaissent ensuite les preuves interactives, introduites à la fin des années 1980, dont le principe est de dissocier la production d'une preuve (où l'« intuition » joue un rôle) de sa vérification (une tâche purement mécanique). Cette dissociation a permis un saut conceptuel en basculant de l'idée de vérifier précisément une preuve à celle d'être convaincu par une preuve. Ces recherches ont ainsi permis dès 1985 la découverte des protocoles *zero-knowledge* (voir par ailleurs dans ce dossier), qui sont en quelque sorte le summum de la cryptographie. Enfin, la théorie de l'approximation algorithmique des problèmes d'optimisation où le théorème PCP a permis la démonstration de résultats d'inapproximabilité optimaux (à moins que P = NP).

Avec ces résultats, l'informatique est sans doute la science qui a, durant la seconde moitié du XX^{e} siècle, le plus bousculé nos croyances les plus refoulées, en particulier sur le rôle du hasard.

Dans une preuve vérifiable avec du hasard (voir en encadré), le vérifieur est totalement libre d'interpréter la preuve π sous la forme qu'il souhaite. C'est lui qui donne un sens aux *bits* de π. À titre d'exemple, voici un PCP$(0, n)$-vérifieur pour la propriété 3COL : « être coloriable avec trois couleurs », portant sur les graphes G à n sommets et m arêtes, le vérifieur prend en entrée la description du graphe G et considère que π encode la liste de $O(n)$ *bits* des couleurs des n sommets. Il vérifie alors en temps $O(n)$ que les couleurs des sommets appartiennent à $\{1, 2, 3\}$, et en temps $O(m)$ que les extrémités de chaque arête ont bien des couleurs différentes. Le vérifieur n'utilise aucun *bit* aléatoire ($r(k) = 0$) et lit $q(k) = O(n)$ *bits* de la preuve π. Comme la propriété 3COL est NP-complète, on en déduit que toute la classe NP appartient à PCP(0, polynôme(k)).

S'aider du hasard

PCP signifie *probabilistically checkable proofs*, soit « preuves vérifiables avec du hasard ». Formellement, un PCP$(r(k), q(k))$-vérifieur d'une propriété $\prod$ est un algorithme V qui :

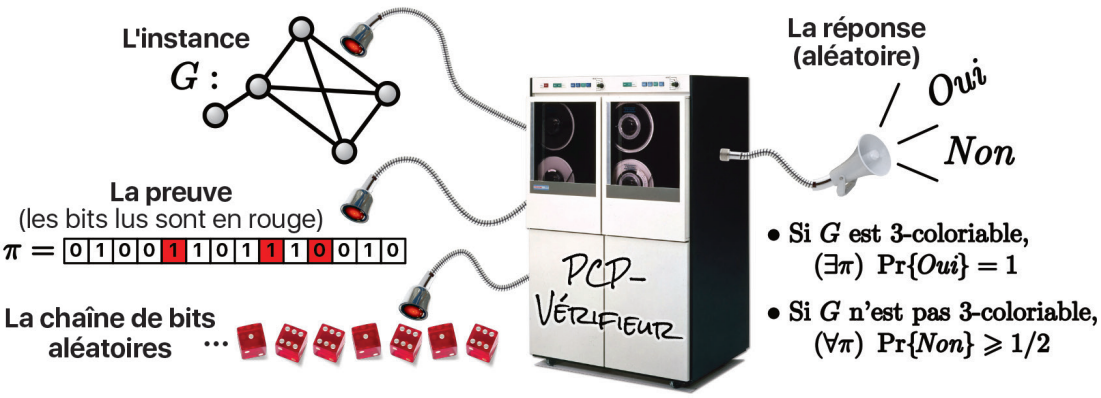
- possède trois entrées : une instance G dont la longueur de l'écriture en *bits* sera notée k, une preuve π censée démontrer que G vérifie bien la propriété $\prod$, et une chaîne de *bits* aléatoires ρ de longueur $r(k)$;
- à la lecture de l'entrée G, utilise les $r(k)$ *bits* aléatoires dans ρ pour choisir $q(k)$ positions à lire dans π ;
- répond « Accepte » ou « Rejette » en fonction des $q(k)$ *bits* lus dans la preuve proposée π ;

le tout en un temps borné par un polynôme en k.

En outre, on demande que la probabilité d'acceptation de V, prise sur l'ensemble des valeurs possibles pour la chaine aléatoire ρ, vérifie :

- si G satisfait la propriété $\prod$, alors il doit exister une preuve π telle que le vérifieur V accepte l'entrée (G, π, ρ) quelle que soit la valeur de ρ (V accepte (G, π) avec probabilité 1). Ainsi, V ne rejette jamais une instance qui satisfait la propriété $\prod$;
- si G ne satisfait pas la propriété $\prod$, alors, quelle que soit la preuve proposée π, le vérifieur V doit rejeter au moins la moitié des entrées (G, π, ρ) (V accepte (G, π) avec une probabilité inférieure ou égale à 1/2). Ainsi, V rejette toute preuve d'une instance ne satisfaisant pas la propriété avec une probabilité au moins 1/2.

La classe PCP$(r(k), q(k))$ est l'ensemble des propriétés $\prod$ ayant un $(O(r(k)), O(q(k)))$-vérifieur, en utilisant les notations de Landau (voir le dossier précédent).



Le théorème PCP énonce que l'on peut trouver un vérifieur bien plus efficace pour la propriété 3COL, en utilisant la puissance du hasard contenue dans la chaîne aléatoire ignorée par le vérifieur naïf précédent. En effet, on peut se contenter de ne lire qu'un nombre *constant* de *bits* de la preuve fournie π (nombre indépendant de la taille de G !) pour se convaincre que G est 3-coloriable ou non.

Des preuves interactives

La classe NP est l'ensemble des propriétés qui admettent des preuves vérifiables de façon déterministe en un temps acceptable (c'est-à-dire borné par un polynôme en la taille k de l'instance). La plupart des problèmes concrets auxquels on se trouve confronté appartiennent à cette classe (comme pour un pays de posséder un chemin de longueur inférieure à 1 500 km passant par toutes les villes de plus de 5 000 habitants). D'après le théorème PCP, il n'y a que deux cas possibles : soit l'instance proposée satisfait la propriété et l'on peut rédiger une preuve π de longueur polynomiale (puisque au plus $q(k)2^{r(k)} = O(1)2^{O(\log(k))}$ = polynôme(k) *bits* seront lus par le vérifieur) que le vérifieur acceptera à coup sûr ; soit G ne satisfait pas la propriété et, quelle que soit la preuve soumise π, le vérifieur la rejettera avec probabilité au moins 1/2 (seuil que l'on peut réduire à volonté en menant plusieurs exécutions du vérifieur sur des chaînes aléatoires indépendantes).

C'est en optimisation que le théorème PCP a eu les conséquences les plus spectaculaires. En effet, on sait depuis le début des années 1970 que la résolution exacte de la quasi-totalité des problèmes utiles d'optimisation est NP-complète, mais en pratique une solution approchée suffirait largement. Or, jusqu'au début des années 1990, la plupart de ces problèmes semblaient résister à être approchés efficacement, avec des facteurs d'erreur allant d'une constante (pour le problème de la coupe maximale) à un polynôme en la taille du graphe (pour la recherche d'une clique de taille maximale). Le théorème PCP a permis de démontrer qu'en fait l'approximation de la plupart de ces problèmes était elle-même NP-complète, donc impossible à moins que P = NP. Les bornes obtenues sur les meilleures approximations possibles s'accordent avec les meilleurs algorithmes d'approximation connus pour de nombreux problèmes clés.

Par la suite, le théorème PCP a été raffiné et de nombreux résultats d'inapproximabilité en ont découlé. Par exemple, Johan Håstad a démontré en 1997 que le problème classique 3SAT admet un PCP-vérifieur ne lisant que trois *bits* de la preuve (et c'est optimal !), ce qui signifie que le problème Max-3SAT n'admet pas de $(7/8 + \varepsilon)$-approximation pour aucun $\varepsilon > 0$. Or, 7/8 est aussi, précisément, le facteur d'approximation du meilleur algorithme connu !

La démonstration moderne du théorème PCP s'effectue en deux temps (voir les deux articles qui suivent). Dans un premier temps, on démontre une forme dégradée de la version preuve interactive en prouvant que l'on peut écrire une preuve de taille exponentielle mais vérifiable en ne lisant qu'un nombre constant de *bits*. Le second temps démontre la version optimisation du théorème, en prouvant itérativement qu'il n'existe pas de c-approximation pour Max-qSAT pour un certain q, pour des constantes c de plus en plus petites. La première étape intervient à un moment crucial de ce processus et est utilisée sur des instances de tailles constantes, où la longueur de la preuve n'a donc pas d'importance ($2^{\text{constante}}$ = une constante), et où seul importe le nombre de *bits* lus.

N.S.

Théorème PCP et inapproximabilité

En 1991, la connexion entre preuves interactives et inapproximabilité est établie, quelques mois avant la démonstration du théorème PCP. Le principe est le suivant : on peut voir l'exécution du PCP(r, q)-vérifieur sur une instance G donnée comme l'évaluation d'une fonction booléenne $f_\rho(\pi)$ dépendant de q *bits* seulement de la preuve π, choisie uniformément au hasard parmi 2^r fonctions booléennes possibles (une par chaîne aléatoire $\rho \in \{0, 1\}^r$). Ainsi, le théorème PCP peut être vu comme la réécriture de la satisfaction d'une propriété $\prod$ par une instance G sous la forme d'un ensemble de taille polynomiale ($2^{r(k)} = 2^{O(\log k)}$ = polynôme(k)) de fonctions booléennes (f_ρ) possédant un nombre constant, q, d'arguments et telles que : si G satisfait la propriété $\prod$, alors il existe une preuve π telle que toutes les fonctions f_ρ s'évaluent à 1 simultanément, sinon au moins la moitié d'entre elles s'évaluent à 0, quelle que soit la preuve fournie π. Déterminer si G vérifie la propriété se ramène donc à décider s'il existe une preuve π qui satisfasse la totalité des fonctions, ou bien si la « meilleure tentative de preuve possible » ne satisfait qu'au plus la moitié d'entre elles. Considérons donc le problème de maximiser le nombre de fonctions satisfaites. Si nous avions une $(1/2 + \varepsilon)$-approximation pour ce problème de maximisation (c'est-à-dire un algorithme qui, en temps polynomial, renvoie une solution qui satisfait à coup sûr au moins une fraction $1/2 + \varepsilon$ du nombre maximum de fonctions booléennes satisfiables simultanément), nous pourrions l'utiliser pour distinguer entre les deux types d'instances G. En effet, pour les instances entièrement satisfiables, un tel algorithme renverrait à coup sûr une solution satisfaisant une fraction $1/2 + \varepsilon$ des fonctions, soit strictement plus que la moitié, ce qui est impossible pour l'autre type d'instance.
Ainsi, le théorème PCP implique qu'il n'existe pas de tel algorithme d'approximation pour ce problème, à moins que P = NP. En fait, cette implication est une équivalence, et c'est d'ailleurs cette seconde version qui sera montrée dans le dernier article de ce dossier.