

Chapitre 14

Protocoles interactifs

14.1 Preuves interactives

14.1.1 Discussion

Les systèmes de preuve interactifs donnent un moyen de définir une version probabiliste de la classe NP, comme les algorithmes probabilistes permettent de définir l'analogie probabiliste de la classe P.

L'étude des systèmes de preuve interactifs a eu de nombreuses conséquences en théorie de la complexité.

Les langages de NP sont ceux pour lesquels, étant donné un mot, son appartenance au langage peut être testée facilement par l'existence d'un certificat court : reformulons cette affirmation en considérant deux entités : un *prouveur* qui produit une preuve d'appartenance, et un *vérificateur* qui la vérifie. Considérons que le prouveur cherche à convaincre le vérificateur de l'appartenance d'un mot donné w au langage. On suppose que le vérificateur correspond à un algorithme en temps polynomial, sinon il pourrait trouver la réponse lui-même. Par contre, on n'impose aucune contrainte sur le prouveur, puisque produire une preuve peut être difficile.

Considérons le problème SAT par exemple. Un prouveur peut convaincre un vérificateur qu'une formule ϕ est satisfiable en fournissant une affectation de ses variables sur laquelle la formule s'évalue en vrai.

Un prouveur peut-il convaincre un vérificateur qu'une formule n'est pas satisfiable ?

Le complémentaire de SAT n'est pas connu comme étant dans NP (et est coNP-complet) et donc on ne peut pas s'attendre à utiliser l'idée de l'existence d'un certificat court (sauf si $P = NP$). Cependant, il s'avère que oui, si l'on ajoute au prouveur et au vérificateur deux caractéristiques supplémentaires : premièrement, ils travaillent en fait en dialoguant. Deuxièmement, le vérificateur peut être un algorithme *probabiliste* qui doit simplement produire la bonne réponse avec forte probabilité, plutôt qu'avec certitude.

Un tel système de prouveur et de vérificateur constitue un *système de preuve interactif*.

14.1.2 Illustration

Comme exemple intuitif de la puissance de la combinaison de l'utilisation de randomisation et d'interactions, considérons le scénario suivant : Maria possède une chaussette rouge et une chaussette jaune, mais son ami Arthur, qui est daltonien, ne croit pas que ses chaussettes sont de couleur distincte. Comment peut-elle le convaincre ?

Voici une méthode : Maria donne les deux chaussettes à Arthur, lui dit laquelle est rouge, laquelle est jaune, et Arthur prend la chaussette rouge dans sa main droite, la chaussette jaune dans sa main gauche. Alors Maria tourne le dos à Arthur et Arthur lance une pièce. Si la pièce dit "pile", alors Arthur garde les chaussettes comme elles sont. Si c'est "face", il les échange entre main droite et main gauche. Il demande alors à Maria de deviner s'il a échangé les chaussettes ou pas. Bien entendu, il suffit à Maria de regarder si les chaussettes sont différentes, mais si elles sont identiques, alors Maria ne peut mieux faire qu'avoir une réponse correcte avec une probabilité 1/2. En répétant l'expérience, disons 100 fois, si Maria répond toujours correctement, Arthur peut se convaincre que les chaussettes sont bien de couleur distincte.

Cette idée est celle derrière le fait que l'on peut résoudre le problème du non-isomorphisme de graphe avec un protocole interactif.

14.1.3 Définition formelle

On voit le *vérificateur* comme une fonction V qui calcule le message suivant à transmettre au prouveur à partir de l'historique de la communication jusque là. La fonction V possède trois entrées :

- l'entrée w , dont on cherche à déterminer l'appartenance au langage ;
- un mot y qui code une suite aléatoire : même si cela est équivalent, pour des raisons de commodités dans les définitions, on préfère voir un vérificateur comme prenant en seconde entrée un mot sur $\{0, 1\}$ aléatoire, plutôt qu'un algorithme probabiliste qui ferait des choix aléatoires lors de son exécution ;
- l'historique des échanges de messages jusque là : si les messages m_1, m_2, \dots, m_i ont été échangés jusque là, on note $m_1 \# m_2 \dots \# m_i$ le mot qui code les messages m_1 jusqu'à m_i .

La sortie de la fonction V est soit le message suivant m_{i+1} dans l'interaction, soit "accepter" soit "rejeter", ce qui termine l'interaction. Autrement dit, $V : M^* \times M^* \times M^* \rightarrow M^*$ où M est l'alphabet, et $V(w, y, m_1 \# m_2 \dots \# m_i) = m_{i+1}$ signifie que sur l'entrée w , et la suite de tirages aléatoires y , l'historique de messages m_1, \dots, m_i , le prochain message du vérificateur est m_{i+1} .

Le *prouveur* est lui sans aucune limite de puissance : on le voit comme une fonction P qui prend deux entrées

- l'entrée w

– l'historique des échanges de message jusque là.

$P(w, m_1 \# m_2 \cdots \# m_i) = m_{i+1}$ signifie que le prouveur envoie le message m_{i+1} au vérificateur après avoir échangé les messages m_1, \dots, m_i sur l'entrée w .

On définit ensuite l'interaction entre le prouveur et le vérificateur. Pour une entrée w , on note $V \leftrightarrow P(w, y) = \text{accepter}$ s'il existe une suite de messages $m_1 m_2 \cdots m_k$ telle que

- pour $0 \leq i < k$, où i est pair, $V(w, y, m_1 \# \cdots \# m_i) = m_{i+1}$
- pour $0 < i < k$, où i est impair, $P(w, m_1 \# \cdots \# m_i) = m_{i+1}$
- le dernier message m_k est "accepter".

Pour simplifier la définition de la classe IP, on suppose que la longueur de l'argument y qui code les choix aléatoires et la longueur de chacun des messages m_i échangés restent inférieure à $p(n)$ pour un polynôme p qui ne dépend que du vérificateur. On suppose d'autre part que le nombre total de messages échangés reste inférieur à $p(n)$.

Pour une entrée w de taille n , on définit

$$\Pr(V \leftrightarrow P \text{ accepte } w) = \Pr(V \leftrightarrow P(w, y) = \text{accepter}).$$

Définition 14.1 (Classe IP) *On dit qu'un langage L est dans IP s'il existe une fonction V calculable en temps polynomial, et une fonction P arbitraire telle que pour toute fonction \bar{P} , et pour toute entrée w*

- $w \in L$ implique que $\Pr(V \leftrightarrow P \text{ accepte } w) \geq 2/3$.
- $w \in L$ implique que $\Pr(V \leftrightarrow \bar{P} \text{ accepte } w) \leq 1/3$.

Remarque 14.1 *En fait, comme on peut voir NP comme l'ensemble des théorèmes qui admettent une preuve courte, on peut voir les protocoles interactifs comme ceux qui admettent une preuve interactive courte.*

Remarque 14.2 *On fait l'hypothèse que les choix probabilistes du vérificateur sont privés, et pas accessibles au prouveur.*

Remarque 14.3 *On peut voir P comme quelqu'un qui essaye de convaincre V que l'entrée vérifie une certaine propriété. En acceptant, V indique qu'il est convaincu que la propriété est vraie. V ne doit pas accepter si un intrus \bar{P} s'insère.*

On peut amplifier les probabilités de succès d'un protocole interactif par répétition, comme nous l'avions vu pour les classes probabilistes RP et BPP, pour rendre les probabilités d'erreur exponentiellement faibles.

Par définition, on obtient directement.

Corollaire 14.1 $\text{NP} \subset \text{IP}$.

Démonstration: Soit $L \in \text{NP}$. On a $w \in L$ si et seulement s'il existe un certificat $u \in M^*$ tel que $\langle w, u \rangle \in A$, pour un problème polynomial A . Il suffit d'un protocole en un échange : sur l'entrée w , le prouveur fournit le certificat u

pour une entrée w dans le langage, et n'importe quoi pour une entrée qui n'est pas dans le langage. Puisque le prouveur n'est pas supposé avoir une puissance restreinte, il peut toujours trouver un certificat u tel que $\langle w, u \rangle \in A$ lorsque $w \in L$. Le vérificateur détermine alors si $\langle w, u \rangle \in A$. Si c'est le cas, il accepte, sinon, il rejette. Le vérificateur accepte donc les mots de L avec probabilité 1, et les mots du complémentaire de L avec probabilité 0, puisqu'aucun prouveur ne peut convaincre le vérificateur. \square

Trivialement :

Corollaire 14.2 $BPP \subset IP$.

On verra en fait que $IP = PSPACE$.

14.1.4 Exemple : non-isomorphisme de graphes

On présente un exemple de problème dans IP qui n'est pas connu pour être dans NP . Étant donné un graphe G , pour le représenter, on a souvent besoin de numéroter ses sommets. On dit que deux graphes G_1 et G_2 sont isomorphes s'ils sont identiques à une renumérotation des sommets près. Si l'on préfère, s'il existe une permutation π des numéros des sommets de G_1 telle que $\pi(G_1) = G_2$, où $\pi(G)$ désigne le graphe obtenu en remplaçant le numéro en chaque sommet par son image par la permutation π . Le problème *de l'isomorphisme de graphes* consiste, étant donnés deux graphes G_1 et G_2 à déterminer s'ils sont isomorphes.

Clairement, le problème de l'isomorphisme de graphe est dans NP , puisque l'on peut prendre comme certificat la description de la permutation π . La question de savoir si ce problème est NP -complet est une question ouverte, qui a été relativement abondamment discutée dans la littérature.

En fait, on peut montrer que le non-isomorphisme de graphe est dans IP , puisqu'on peut déterminer si deux graphes ne sont pas isomorphes.

Le protocole IP est le suivant :

- V choisit $i \in \{1, 2\}$ au hasard de façon uniforme. On permute les sommets de G_i pour obtenir un nouveau graphe H . V envoie H à P .
- P identifie lequel de G_1 et de G_2 a été utilisé pour produire H . Soit G_j ce graphe. P envoie j à V .
- V accepte si $i = j$. Rejette sinon.

Pour se convaincre qu'il s'agit d'un protocole interactif correct, observons que si G_1 n'est pas isomorphe à G_2 , alors il existe un prouveur tel que V accepte avec probabilité 1, parce que les graphes ne sont pas isomorphes.

Sinon, si G_1 et G_2 sont isomorphes, alors le mieux qu'un prouveur puisse faire est de deviner parce qu'une permutation de G_1 ressemble exactement à une permutation de G_2 . Donc dans ce cas, pour tout prouveur la probabilité que V accepte est $\leq 1/2$.

14.1.5 $IP = PSPACE$

Théorème 14.1

$IP = PSPACE$.

14.2 Vérification probabiliste de preuve

14.2.1 Définition

On considère un modèle qui est essentiellement le même que précédemment, si ce n'est que qu'on n'autorise pas les erreurs dans le cas positif.

Définition 14.2 (Classe PCP) *On dit qu'un langage L est dans PCP s'il existe une fonction V calculable en temps polynomial, et une fonction P arbitraire telle que pour toute fonction \overline{P} , et pour toute entrée w*

- $w \in L$ implique que $\Pr(V \leftrightarrow P \text{ accepte } w) = 1$.
- $w \in L$ implique que $\Pr(V \leftrightarrow \overline{P} \text{ accepte } w) \leq 1/2$.

Encore une fois, en utilisant des répétitions (réduction de l'erreur), la constante $1/2$ est arbitraire, on pourrait prendre n'importe quel $\epsilon > 0$.

On peut supposer que le vérificateur inclue l'entrée w et les précédents messages m_1, \dots, m_i dans chacune de ses requêtes à P , et que la réponse de P est un unique bit : en faisant comme cela, on peut voir P comme un *oracle*, c'est-à-dire comme un langage.

On présente dans la littérature souvent PCP en présentant l'oracle comme détenant une preuve que l'entrée w est dans L , et les requêtes comme l'extraction de bits de cette preuve.

On place par ailleurs généralement des bornes sur le nombre de bits aléatoires que V utilise, et le nombre de requêtes à l'oracle qu'il peut faire.

On dit qu'un protocole interactif est $(r(n), q(n))$ -borné si sur toute entrée de taille n , le vérificateur utilise moins de $r(n)$ bits aléatoires, et fait au plus $q(n)$ requêtes à son oracle. On note $\text{PCP}(r(n), q(n))$ la famille des langages reconnaissables par un protocole $(\mathcal{O}(r(n)), \mathcal{O}(q(n)))$ -borné.

14.2.2 $\text{NP} = \text{PCP}(\log n, 1)$

Le théorème suivant est très puissant, et est la conséquence de toute une lignée de travaux scientifiques :

Théorème 14.2

$$\text{NP} = \text{PCP}(\log n, 1).$$

La preuve du théorème n'est pas facile.

Ce théorème dit que les problèmes de NP possèdent des protocoles interactifs qui utilisent au plus $\mathcal{O}(\log n)$ bits aléatoires, et qui utilisent au plus un nombre constant, indépendant de la taille de l'entrée, de bits de la preuve.

14.2.3 Conséquences sur la théorie de l'approximation

Le théorème possède toutefois des conséquences directes sur la théorie de l'approximation. On a vu des problèmes variés de décision comme par exemple 3SAT qui sont NP-complets. Souvent à ces problèmes est associé un problème d'optimisation qui peut être un problème de maximisation ou de minimisation.

Par exemple, le problème MAX3SAT demande de produire une affectation des variables d'une formule ϕ en forme normale conjonctive avec trois littéraux par clause (en 3CNF) qui maximise le nombre de clauses satisfaites.

Un α -algorithme d'approximation pour un problème d'optimisation est un algorithme qui produit une réponse qui reste dans un facteur constant α de l'optimal. Pour un problème comme MAX3SAT, cela signifie produire une réponse qui rend le nombre de clauses satisfaites au moins α multiplié par le maximum, pour $0 < \alpha \leq 1$, indépendamment de la taille de l'entrée. La constante α se nomme le rapport d'approximation.

Proposition 14.1 *Il existe un algorithme en temps polynomial qui, étant donné une formule ϕ en 3CNF avec n variables et m clauses, telle que chaque clause possède trois variables distinctes, produit une affectation qui satisfait $7m/8$ clauses.*

Théorème 14.3 *Il existe un α -algorithme polynomial d'approximation pour MAX3SAT si et seulement si $P = NP$.*

14.3 Notes bibliographiques

Ce chapitre a été rédigé en reprenant essentiellement [Sipser, 1997], [Kozen, 2006] et [Arora and Barak, 2009].

Bibliographie

- [Arora and Barak, 2009] Arora, S. and Barak, B. (2009). *Computational Complexity : A Modern Approach*. Cambridge University Press.
- [Kozen, 2006] Kozen, D. (2006). *Theory of computation*. Springer-Verlag New York Inc.
- [Sipser, 1997] Sipser, M. (1997). *Introduction to the Theory of Computation*. PWS Publishing Company.