

NP - Complétude

Introduction

Tous les algorithmes étudiés étaient des algorithmes à temps polynomial : leurs temps d'exécution au pire des cas est $O(k^n)$ où n est la taille de l'entrée et k est une constante. Mais tous les problèmes ne peuvent être résolus en un temps polynomial. On peut donc distinguer deux types de problèmes :

- **Problèmes faciles** : Tout problème possédant une solution de complexité polynomiale est considéré "facile".
- **Problèmes difficiles** : Une solution exponentielle, ou pire qu'exponentielle, est associée à un problème "difficile".

1 Vocabulaire général

Problème : Question comportant un ou plusieurs paramètres ; par exemple, quel est le plus court chemin entre deux sommets donnés d'un graphe ?

Instance d'un problème : soit X le problème caractérisé par l'ensemble E de ses données, $E = e_1, \dots, e_n$, une instance de X serait alors caractérisée par un ensemble concret $E' = e'_1, \dots, e'_n$, une deuxième instance serait caractérisée par un deuxième ensemble concret $E'' = e''_1, \dots, e''_n$ et ainsi de suite. L'invariant suivant est toujours vérifié : $|E| = |E'| = |E''|$.

$G = (S, A), (x, y) \in G$, quel est le plus court chemin entre x et y dans G ?

Problème abstrait : relation binaire entre un ensemble d'instances d'un problème et un ensemble de solutions à ce problème.

Problème de décision (ou décisionnel) : il possède pour solution oui ou non. On en déduit qu'un problème décisionnel abstrait fait correspondre à toute instance l'ensemble des solutions *vrai, faux*. On représente fréquemment un problème de décision par un ensemble X qui est un sous-ensemble général Y . Alors, pour chaque instance $y \in Y$, le problème consiste à décider si oui ou non $y \in X$.

Problème de calcul : calculer la solution d'un problème. (ex : Calculer le plus court chemin entre deux sommets donnés d'un graphe).

Résoluble en un temps polynomial : il existe un algorithme permettant la résolution en $O(n^k)$ pour une constante k .

Réductibilité : un problème X peut être ramené (réduit) à un autre problème X' si une instance quelconque de X peut être facilement reformulée comme une instance de X' dont la solution sera aussi solution pour X .

Réductibilité en temps polynomial : La fonction qui réduit le problème X en X_0 peut être obtenue en

temps polynomial.

Problème de calcul : calculer la solution d'un problème.

Dans ce qui suit, nous présenterons quatre classes de complexité dont on parle le plus souvent : classe P, classe NP, classer NP-complet et NP-difficile.

Définition 2 : P est la classe de tous les problèmes de décision qui peuvent être résolus par un algorithme polynomial.

P-complet : Un problème décisionnel appartient à ce classement s'il fait partie de la classe P et si tout problème P dans P peut s'y réduire en temps poly-logarithmique en utilisant un ordinateur avec un nombre polynomial de processeurs. En pratique, si X est un problème dans P alors il appartient à P et pour tout problème Y dans P, il existe les constantes c et k telles que Y puisse être réduit en X en $O((\log(n))^c)$ avec un ordinateur qui possède $O(n^k)$ processeurs en parallèle.

Définition 3 : NP (Non-déterministic Polynomial time) est la classe de tous les problèmes de décision dont la solution peut être vérifiée en temps polynomial, i.e. si l'on nous donne une solution certifiée, il est possible de vérifier que cette solution est correcte en un temps polynomial par rapport à la taille de l'entrée.

Exemple :

- Systèmes d'équations linéaires
- Problèmes des nombres premiers
- HAM est le problème qui, étant donné un graphe G, consiste à trouver (s'il existe) un cycle hamiltonien de G.
- HAMD est le problème de décision qui, étant donné un graphe G, consiste à décider si oui ou non G contient un cycle hamiltonien.

Clairement, $P \subseteq NP$ mais la question qui se pose est : $P = NP$?

Définition 4 : Un problème est NP est NP-complet si tout problème NP s'y réduit en temps polynomial.

Autrement, les problèmes NP les plus difficiles sont NP-Complets.

Propriété : Si un seul problème NP-Complet peut être résolu en un temps polynomial, alors tous les problèmes de NP peuvent être résolus en un temps polynomial.

Un problème NP-Complet est un problème dans NP au moins aussi difficile que tout autre problème de NP.

- Il existe une réduction polynomiale qui permet de décrire n'importe quelle instance de n'importe quel problème de NP comme une instance de NP-Complet.
- Si on sait résoudre toutes les instances d'un problème NP-complet on sait résoudre toutes les instances de tous les problèmes NP.

Si un problème NP-Complet est dans la classe $P(NP - Complet \cap P \neq \emptyset)$, alors $P = NP$.

Exemples de problèmes NP-Complet

- Satisfaisabilité d'une formule booléenne (SAT)
- Problème de sac à dos
- Voyageurs de commerce
- Problème du coloriage de graphe

Problème NP-Complet au sens faible :

- Il est NP-Complet
- Il existe un algorithme pseudo-polynomial (peut s'exprimer comme un polynôme des paramètres du problème) pour le résoudre

Problème NP-Complet au sens fort :

- Il appartient à la classe NP
- Il reste NP-Complet même si on limite la taille des paramètres du problème

Définition 5 : Un problème est NP-difficile si tout problème s'y réduit en temps polynomial.

On a $NP\text{-complet} \subseteq NP\text{-difficile}$, par contre un problème NP-difficile n'est pas nécessairement dans NP.

2 Processus de réduction

Soient deux problèmes P1 et P2, réduire P1 en P2 signifie transformer une instance quelconque de P1 en une instance équivalente de P2; ainsi, P2 peut servir de sous-routine pour résoudre P1. On note $P1 \alpha P2$.

Principe : Le processus de réduction donne un moyen de résoudre un problème P1 en temps polynomial de la façon suivante :

1. Etant donné une instance a de P1, utiliser une réduction à temps polynomial pour la transformer en une instance b d'un problème P2.
2. Exécuter l'algorithme de décision à temps polynomial de P2 sur l'instance b,
3. Utiliser la réponse pour b comme réponse pour a.

Définition 6 : Soient A et B, deux problèmes. On dit que A est polynomialement Turing réductible à B (noté $A <_T^P B$) s'il existe un algorithme pour résoudre A qui serait exécutable en temps polynomial si on pouvait résoudre n'importe quelle instance de B en un temps unité.

Théorème : Si $A <_T^P B$ et si B peut être résolu en un temps polynomial, alors A est également résoluble en temps polynomial.

Définition 7 : Un problème de décision A est dit NP-Complet, si $A \in NP$ (1) et $X <_T^P A \forall X \in NP$ (2).

Pour les problèmes NP-difficiles on a la propriété suivante : Un problème est NP-difficile s'il est plus difficile au sens de la réduction de turing qu'un problème NP-Complet. A est NP-difficile s'il existe B NP-Complet tel que $B <_T^P A$.

Conclusion

Personne n'a trouvé d'algorithme polynomial pour un problème NP-Complet. En outre, personne n'a pu prouver qu'il n'en existait pas.