Problème NP-complet

En théorie de la complexité, un **problème NP-complet** (c'est-à-dire un problème complet pour la classe NP) est un problème de décision vérifiant les propriétés suivantes :

- il est possible de vérifier une solution efficacement (en temps polynomial) ; la classe des problèmes vérifiant cette propriété est notée NP ;
- tous les problèmes de la classe NP se ramènent à celui-ci via une réduction polynomiale ; cela signifie que le problème est au moins aussi difficile que tous les autres problèmes de la classe NP.

Un problème **NP-difficile** est un problème qui remplit la seconde condition, et donc peut être dans une classe de problème plus large et donc plus difficile que la classe NP.

Bien qu'on puisse *vérifier* rapidement toute solution proposée d'un problème NP-complet, on ne sait pas en *trouver* efficacement. C'est le cas, par exemple, du problème du voyageur de commerce ou de celui du problème du sac à dos.

Tous les algorithmes connus pour résoudre des problèmes NP-complets ont un temps d'*exécution* exponentiel en la taille des données d'entrée dans le pire des cas, et sont donc inexploitables en pratique même pour des instances de taille modérée.

La seconde propriété de la définition implique que s'il existe un algorithme polynomial pour résoudre un quelconque des problèmes NP-complets, alors tous les problèmes de la classe NP peuvent être résolus en temps polynomial. Trouver un algorithme polynomial pour un problème NP-complet ou prouver qu'il n'en existe pas permettrait de savoir si P = NP ou P ≠ NP, une question ouverte qui fait partie des problèmes non résolus en mathématiques les plus importants à ce jour.

En pratique, les informaticiens et les développeurs sont souvent confrontés à des problèmes NP-complets. Dans ce cas, savoir que le problème sur lequel on travaille est NP-complet est une indication du fait que le problème est difficile à résoudre, donc qu'il vaut mieux chercher des solutions approchées en utilisant des algorithmes d'approximation ou utiliser des heuristiques pour trouver des solutions exactes.

Sommaire

- Définition formelle
- Histoire
- Preuves de NP-complétude
- Liste de problèmes NP-complets
  - Contre-exemples
- Résolution
  - Méthodes exactes
  - Méthodes approchées
- NP-complétude faible
- Notes et références
- Voir aussi
  - Articles connexes
  - Liens externes

Définition formelle

Un problème de décision peut être décrit mathématiquement par un langage formel, dont les mots correspondent aux instances du problème pour lesquelles la réponse est oui. Sans perte de généralité, on peut supposer que tous les langages qu'on considère sont définis sur le même alphabet Σ.

Deux classes de complexité, présentées de manière plus détaillée dans Théorie de la complexité, interviennent dans la définition de la NP-complétude :

- La classe NP est l'ensemble des langages décidables en temps polynomial par une machine de Turing *non déterministe*.
- La classe P est l'ensemble des langages décidables en temps polynomial par une machine de Turing déterministe. Elle est incluse dans NP.

Un **langage** L est **NP-difficile** s'il existe une réduction polynomiale de tout langage $L' \in NP$ à L, c'est-à-dire une fonction $f : \Sigma^* \mapsto \Sigma^*$ (où Σ est l'alphabet de ces langages), calculable en temps polynomial sur une machine de Turing déterministe, telle que :

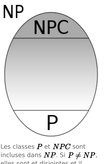
$$\forall w \in \Sigma^*, w \in L' \Leftrightarrow f(w) \in L\,.$$

Si de plus $L \in NP$, alors le **langage** L est **NP-complet**.

La classe des langages NP-complets est notée NPC.

On dit qu'un *problème de décision* est « NP-complet » lorsque le langage correspondant est NP-complet. C'est une notion informelle car il existe plusieurs moyens de coder les instances d'un problème, mais cela ne pose pas de difficultés dans la mesure où on emploie un codage *raisonnable* du problème vers le langage considéré (voir la section NP-complétude faible).

Par abus de langage, on qualifie souvent de « NP-complets » des problèmes d'*optimisation* et non des problèmes de décision. Ainsi, on dira que le problème du voyageur de commerce, qui consiste à trouver un plus court circuit passant une seule fois par chacun des sommets d'un graphe connexe fini, est NP-complet. Cela se justifie par le fait qu'on peut transformer tout problème d'optimisation en problème de décision et réciproquement (voir la section « De l'existence à la décision » de l'article sur la théorie de la complexité).



Les classes P et NPC sont incluses dans NP. Si $P \neq NP$, elles sont et disjointes et il existe des problèmes NP-intermédiaires, i.e. dans $NP - (NPC \cup P)$ (représenté ici par la section médiane de l'ovoïde NP).

Histoire

Le concept de NP-complétude a été introduit en 1971 par Stephen Cook dans une communication intitulée *The complexity of theorem-proving procedures*[1] (La complexité des procédures de démonstration de problèmes), bien que le mot « NP-complet » n'apparaissait pas explicitement dans l'article. Lors de la conférence à laquelle il a été présenté, une discussion acharnée a eu lieu entre les chercheurs présents pour savoir si les problèmes NP-complets pouvaient être résolus en temps polynomial sur machine de Turing déterministe. John Hopcroft a finalement convaincu les participants que la question devait être remise à plus tard, personne n'ayant réussi à démontrer ou infirmer le résultat. [réf. souhaitée]

La question de savoir si P = NP n'est toujours pas résolue. Elle fait partie des problèmes du prix du millénaire, un ensemble de sept problèmes pour lesquels l'Institut de mathématiques Clay offre un prix d'un million de dollars. Il « suffirait » de trouver un seul problème NP qui soit à la fois NP-complet et P pour démontrer cette hypothèse, ou d'exhiber un seul problème NP qui ne soit pas dans P pour démontrer sa négation.

Le résultat de l'article de Cook, démontré de manière indépendante par Leonid Levin en URSS[2], est maintenant connu sous le nom de théorème de Cook-Levin. Ce théorème affirme qu'il existe un problème NP-complet. Cook a choisi le problème SAT et Levin un problème de pavage. En 1972, Richard Karp a prouvé la NP-complétude de plusieurs autres problèmes fondamentaux en informatique et très disparates, connus comme la liste des 21 problèmes NP-complets de Karp. Depuis, on a démontré la NP-complétude de milliers d'autres problèmes.

Diagramme d'Euler pour les problèmes NP-complets.

Preuves de NP-complétude

Pour démontrer la NP-complétude d'un nouveau problème $\Pi_2 \in NP$, la méthode usuelle consiste à trouver un problème NP-complet Π_1 déjà connu qui se réduit à Π_2. cela suffit à démontrer que Π_2 est NP-complet. En effet, pour tout $\Pi \in NP$, on a $\Pi \propto_\mathrm{P} \Pi_1 \propto_\mathrm{P} \Pi_2$ (la relation de réduction polynomiale $\propto_\mathrm{P}$ est transitive).

Pour illustrer cette méthode, voici une démonstration de la NP-complétude de l'optimisation linéaire en nombres entiers (OLNE) par une réduction à partir de SAT, qui est NP-complet comme expliqué plus haut.

L'OLNE consiste à déterminer si un système d'inéquations linéaires a au moins une solution entière.

Exemple

$2y \geq 1,\ 2y - 4x \leq 1,\ 2y + 4x \leq 6$ a une solution entière ($x = 1,\ y = 1$) ;
$2y \geq 1,\ 2y - 4x \leq 1,\ 2y + 4x \leq 5$ n'a aucune solution entière.

SAT consiste à déterminer si une formule logique sous forme normale conjonctive est satisfiable.

Exemple

$(v_1 \vee v_2) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_1 \vee \neg v_2)$ est satisfiable en posant v_1=vrai et v_2=faux.
$(v_1 \vee v_2) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_1)$ n'est pas satisfiable.

Le problème de l'OLNE est bien un problème NP : vérifier qu'une assignation des variables est solution est réalisable efficacement[3]. On définit la fonction de réduction f de la façon suivante. Soit $\boldsymbol{\phi}$ une formule sous forme normale conjonctive. On définit $\boldsymbol{f(\phi)}$ comme étant le problème d'OLNE suivant :

- A chaque variable v_i correspond une variable x_i du système. De plus, on ajoute au système la contrainte $0 \leq x_i \leq 1$.
- Pour chaque clause de la forme $a_1 \vee \cdots \vee v_k \vee \neg v_{k+1} \vee \cdots \vee \neg v_n$, avec $v_1, \ldots, v_n$ des variables pas forcément distinctes, on ajoute au système la contrainte $x_1 + \cdots + x_k + (1 - x_{k+1}) + \cdots + (1 - x_n) \geq 1$.

Exemple

Pour la formule $(v_1 \vee v_2) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_1 \vee \neg v_2)$, le système linéaire contient cinq inéquations :
$0 \leq x_1 \leq 1,\ 0 \leq x_2 \leq 1,\ x_1 + x_2 \geq 1,\ x_1 + (1 - x_2) \geq 1$ et $(1 - x_1) + (1 - x_2) \geq 1$.

On constate alors que si $\boldsymbol{\phi}$ est satisfiable, alors en posant :

- $x_i = 0$ si v_i=faux,
- $x_i = 1$ si v_i=vrai,

on obtient une solution du problème d'OLNE. Réciproquement, si le problème d'OLNE a une solution, alors la formule peut être satisfaite en mettant à vrai les variables v_i telles que $x_i = 1$. Ainsi, la fonction f est bien une réduction de SAT à l'OLNE. De plus, f est calculable en temps polynomial donc l'OLNE est un problème NP-complet.

Liste de problèmes NP-complets

Parmi les problèmes NP-complets notables, on peut citer :

- SAT et sa variante 3-SAT ;
- le problème du cycle hamiltonien et sa variante pondérée, le problème du voyageur de commerce ;
- la clique maximum (équivalent au stable maximum et à la couverture de sommets minimum) ;
- le problème ensemble dominant minimal ;
- les problèmes de coloration de graphe ;
- le problème du sac à dos.

Il est à noter qu'il s'agit alors d'un abus de langage de parler de problème NP pour certains de ces problèmes, car ils ne sont pas des problèmes de décision. Cependant, les problèmes de décision associés à un problème d'optimisation, typiquement de la forme : « existe-t-il une solution de coût inférieur à… ? », sont eux bien NP-complets.

Le schéma de droite indique, pour une partie de ces problèmes, dans quel ordre on prouve en général la NP-complétude. On notera que cet ordre de réduction n'est pas le seul possible, puisque par définition, il existe toujours une réduction entre deux problèmes NP-complets quelconques. Son intérêt est de simplifier les démonstrations.

Chaîne des réductions communément utilisées pour prouver la NP-complétude de quelques problèmes classiques.

Contre-exemples

Parfois, une modification mineure transforme un problème NP-complet en problème de la classe P. Quelques exemples :

- 3-SAT, restriction de SAT au cas où toutes les clauses ont au plus trois variables, est NP-complet tandis que 2-SAT peut être décidé en temps linéaire ;
- le problème de coloration de graphe est NP-complet avec trois couleurs, mais polynomial avec deux couleurs ;
- savoir s'il existe un circuit hamiltonien est NP-complet tandis que savoir s'il existe un circuit eulérien est dans *P*.

Certains problèmes dans NP, par exemple ceux exploités en cryptographie à clé publique, sont supposés être strictement entre P et NPC. En particulier, la méthode RSA repose sur la difficulté de la factorisation des entiers. Aucun algorithme polynomial n'est connu pour le résoudre, mais le problème appartient à NP∩co-NP et n'est donc vraisemblablement pas NP-complet.

À l'opposé, il existe des problèmes de plus difficiles que les problèmes NP-complets. Certains problèmes sont indécidables, par exemple le problème de l'arrêt. Certains problèmes décidables nécessitent un espace exponentiel et ne sont donc pas dans NP, par exemple la comparaison des langages dénotés par deux expressions rationnelles dans lesquelles on autorise, en plus des opérations usuelles, l'opération *carré*[4] (c'est un problème complet dans EXPSPACE).

Résolution

Méthodes exactes

La NP-complétude ne donne une idée de la difficulté d'un problème que dans le pire cas. Autrement dit, même si on ne sait pas résoudre rapidement toutes les instances d'un problème NP-complet, c'est parfois possible pour une partie d'entre elles.

Ainsi, des algorithmes par séparation et évaluation, non polynomiaux, peuvent avoir un temps d'exécution raisonnable pour des instances relativement grandes.

Si on a besoin de résoudre seulement une sous-classe des instances possibles, alors le problème peut devenir facile. Par exemple, la recherche de l'indépendant maximum (NP-complet) est possible en temps polynomial sur les graphes bipartis ou d'autres familles de graphes.

Dans certains cas, on sait démontrer que les instances *aléatoires* d'un problèmes NP-complets sont faciles. À titre d'exemple :

- Le coloriage de graphe avec k couleurs ($k \geq 3$) est NP-complet, mais les instances difficiles correspondent à des graphes peu denses. Ainsi, si on choisit un graphe à n sommet aléatoirement, de façon uniforme parmi les $2^{n(n-1)/2}$ possibles, la probabilité d'existence d'un k-coloriage tend vers 0 quand n augmente. De plus, un algorithme de coloriage glouton, explorant les sommets dans un ordre arbitraire, termine après avoir exploré O(1) sommets en moyenne[5] ;
- À l'opposé, un graphe aléatoire contient un circuit hamiltonien avec une probabilité tendant vers 1 quand sa taille tend vers l'infini[6], et on peut trouver un tel circuit en temps probabiliste polynomial[7].

Enfin, l'approche de la complexité paramétrée consiste à identifier un paramètre qui, dans le cas où il reste petit, permet de résoudre rapidement le problème. En particulier les algorithmes FPT en un paramètre k permettent de résoudre un problème en temps proportionnel au produit d'une fonction quelconque de k et d'un polynôme en la taille de l'instance, ce qui fournit un algorithme polynomial quand k est fixé.

Méthodes approchées

Lorsque les méthodes de résolution exactes ne sont pas applicables, des algorithmes d'approximations (ou à défaut des heuristiques) permettent de trouver des solutions approchées de l'optimum en un temps raisonnable pour un certain nombre de problèmes NP-complets.

- Certains problèmes sont inapproximables, comme le problème du voyageur de commerce.
- Par contre, le problème du voyageur de commerce restreint au cas où les distances satisfont l'inégalité triangulaire est approximable à un facteur 3/2 : il existe un algorithme polynomial qui calcule un chemin dont la longueur est au plus 3/2 fois plus grande que celle du chemin optimum.
- D'autres problèmes admettent un schéma d'approximation polynomial, c'est-à-dire qu'ils sont approximables à un facteur $(1+\epsilon)$ en temps polynomial pour tout $\epsilon > 0$. C'est le cas du problème du sac à dos.

NP-complétude faible

Comme mentionné plus haut, il existe plusieurs manières de coder les instances d'un problème sous forme de mots (faits d'une suite finie de symboles d'un alphabet déterminé et fini). Choisir un encodage particulièrement long diminue artificiellement la complexité du problème, puisque celle-ci est exprimée en fonction de la taille de l'entrée.

En particulier, un nombre entier n peut être encodé sur $\log_a n$ symboles s'il est écrit en base a (ou une base supérieure) mais occupe n symboles s'il est écrit en unaire, ce qui est exponentiellement plus grand.

Les problèmes qu'on peut résoudre en temps polynomial lorsque leurs entrées sont codées en unaire sont appelés problèmes « NP-complets faibles ». Le problème du sac à dos et ses variantes appartiennent à cette catégorie. En effet, ils peuvent être résolus par programmation dynamique en temps polynomial en le nombre d'objets et la plus grande quantité intervenant dans l'entrée.

Par opposition, les problèmes qui restent NP-complets dans ce cas sont dits *forts*. C'est le cas de la plupart des autres problèmes cités dans cet article, comme SAT, CLIQUE, la coloration de graphe, etc.

Notes et références

1. Stephen Cook, *The complexity of theorem proving procedures*, Proceedings of the Third Annual ACM Symposium on Theory of Computing, pages 151–158, 1971.
2. À cette époque les échanges scientifiques entre le monde occidental et l'URSS étaient très ténus.
3. Pour être précis, la taille de la solution doit être bornée polynomialement en la taille de l'entrée, ce qui est toujours possible ici.
4. Albert R. Meyer et Larry Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential space (http://people.csail.mit.edu/meyer/rsq.pdf), *13th IEEE Symposium on Switching and Automata Theory*, Oct 1972, pp. 125-129.
5. Une démonstration est donnée dans l'ouvrage de Wilf cité plus haut.
6. Jean-Paul Delahaye, L'intelligence et le calcul, Belin - Pour la Science, 2002, (ISBN 2-84245-040-X) (chapitre 1, « Les lois de tout ou rien »).
7. D. Angluin and L. G. Valiant, *Fast probabilistic algorithms for hamiltonian circuits and matchings*, Journal of Computer and System Sciences (**en**), Volume 18, Issue 2, pp. 155-193, 1979.

Voir aussi

Articles connexes

- #P-complet
- Liste de problèmes NP-complets
- Problème P = NP
- Théorème de Cook
- Théorie de la complexité des algorithmes
- Conjecture des jeux uniques
- Kernelisation

Liens externes

- (**en**) Version en ligne du livre *Algorithms and Complexity* de Herbert S. Wilf (http://www.math.upenn.edu/~wilf/AlgComp3.html) (contient un chapitre sur les problèmes NP-complets).
- (**en**) A compendium of NP optimization problems (http://www.csc.kth.se/~viggo/problemlist/) (résultats d'approximabilité sur les problèmes NP-complets).
- (**en**) La classe NPC sur le Complexity Zoo (https://complexityzoo.uwaterloo.ca/Complexity_Zoo:NP:npc)

Ce document provient de « https://fr.wikipedia.org/w/index.php?title=Problème_NP-complet&oldid=153156566 ».

La dernière modification de cette page a été faite le 18 octobre 2018 à 10:49.

Droit d'auteur : les textes sont disponibles sous licence Creative Commons attribution, partage dans les mêmes conditions ; d'autres conditions peuvent s'appliquer. Voyez les conditions d'utilisation pour plus de détails, ainsi que les crédits graphiques. En cas de réutilisation des textes de cette page, voyez comment citer les auteurs et mentionner la licence.
Wikipedia® est une marque déposée de la Wikimedia Foundation, Inc., organisation de bienfaisance régie par le paragraphe 501(c)(3) du code fiscal des États-Unis.