VITAMINES

A Complexité

Les organismes qui soutiennent uniquement les recherches ne présentant aucun risque et aucune possibilité d'erreur finiront à la longue par n'être que des soutiens de la médiocrité. Si nous savons soutenir avec sagesse et courage les marginaux qui effectuent des travaux que l'orthodoxie considère insensés ou dépourvus d'intérêt, peut-être sauverons nous de temps en temps, au bénéfice de la science, un Sophus Lie ou un Hermann Grassmann, des gens dont les idées resteront encore célèbres bien après que s'est éteinte notre excitation pour la mode du moment.

FREEMAN DYSON*

LA complexité.

A.1 Notions de complexité

Très souvent on ramène la mesure de la complexité d'un problème, à la mesure du nombre d'instructions que possède le plus petit algorithme capable d'obtenir la solution.

Mais le premier type de complexité que nous aborderons ici est celui de la *complexité temporelle*. C'est le temps que met un algorithme donné pour résoudre un certain problème.

Si n est la taille des données d'entrée (la taille d'une instance du problème), alors on dit qu'un problème est de complexité $\mathbf{O}(f(n))$ si le nombre maximum d'étapes nécessaires à sa résolution est majoré pour n supérieur à un certain n_0 par $f(n)$.

Un algorithme est dit *polynomial* si f est un polynome de degré égal à la taille n du problème. Ces problèmes seront réputés faciles.

Un algorithme est dit *exponentiel* si la complexité ne peut être majorée par une fonction polynomiale. Le tableau A.1 montre ce que cela signifie concrétement lorsqu'on utilise une machine qui prend 1μs pour résoudre une instance de taille 1 du problème.

** D'Éros à Gaïa, Seuil (1995).*

complexité	taille n d'une instance				
	1	5	10	50	100
n	10^{-6}	5.10^{-6}	10^{-5}	5.10^{-5}	10^{-4}
n^2	10^{-6}	2.510^{-5}	10^{-4}	2.510^{-3}	10^{-2}
n^3	10^{-6}	1.2510^{-4}	10^{-3}	0.125	1
n^4	10^{-6}	6.2510^{-4}	10^{-2}	6.25	100
n^5	10^{-6}	310^{-3}	0.1	313	10^4
e^2	10^{-6}	310^{-5}	10^{-3}	1.110^9	1.310^{24}
e^3	10^{-6}	210^{-4}	0.059	7.210^{17}	5.210^{41}

TAB. A.1 – Durée du calcul, en seconde, pour des problèmes polynomiaux et exponentiels en fonction de la taille du problème

A.2 La classe P

Un problème appartient à la classe P s'il existe un algorithme déterministe[1] permettant de résoudre toute instance de ce problème et ce, en un temps polynomial.

L'ensemble des problèmes de la classe P contient les problèmes dont le temps de résolution est borné par une fonction polynomiale. C'est la classe des problèmes considérés comme faciles.

Un problème n'appartient à la classe P que si aucun cas particulier du problème ne requiert un temps de résolution plus que polynomial. Autrement dit, la méthode de résolution du problème est déterministe en ce sens qu'elle assure l'obtention d'une solution en un temps inférieur à une puissance donnée de la taille N du problème.

On peut également définir une machine de Turing non déterministe qui résoud un problème par essais succéssifs. Par exemple, pour voir si un entier est premier ou non, la machine non déterministe choisit un diviseur possible au hasard, effectue la division et, si le reste est nul elle conclut que le nombre donné au départ n'est pas premier. En revanche, la machine déterministe cherchera un diviseur par un procédé systématique.

Dans le cas d'une machine non déterministe, le temps nécessaire à la résolutiion d'un problème est donné par la longueur du calcul le plus court possible ; ainsi les machines non déterministes semblent avoir un avantage énorme sur les machines déterministes car il est plus facile de vérifier que l'on a trouvé une solution que de la chercher.

Cependant, personne n'a jusqu'ici été capable de démontrer que les problèmes que l'on peut résoudre en un temps polynomial grâce à une machine non déterministe (on désigne la classe de ces problèmes par NP) sont intrinsèquement plus difficile que ceux de la classe P. Le problème de savoir si la classe P est distincte de la NP est connu sous le nom de problème $P - NP$; il est devenu une des principales énigmes des mathématiques contemporaines.

En 1970, Stephen Kook, de l'Université de Toronto, a réalisé d'importants travaux sur le sujet. Il s'intéressait aux conditions dans lesquelles une proposition logique complexe est vraie. Par exemple, la proposition comlexe formée lorsque l'on relie deux propositions simples par le mot « ou » est vraie si l'une des deux propositions simples ou les deux à la fois sont vraies. Il est en général difficile de déterminer les domaines des valeurs de vérité pour lesquels la proposition complexe est vraie. S. Cook a démontré que ce problème de l'évaluation (appelé aussi problème de la satisfiabilité), est aussi difficile que toout autre problème de la classe NP. En d'autres termes,

[1] un algorithme est dit déterministe si, après avoir exécuté une instruction, il n'y a pas de liberté sur le choix de l'instruction suivante à effectuer

il existe un algorithme efficace pour résoudre le problème de l'évaluation si et seulement si il en existe un pour tous les autres problèmes de la classe NP. Un problème qui possède cette propriété universelle par rapport à une classe de problèmes est dit complet pour cette classe.

Il fallut un an pour se rendre compte de l'importance du résultat de Cook. En 1971, Richard Karp, de l'Université de Berkeley, voulut savoir quels autres problèmes naturels jouaient le même rôle vis-à-vis de la classe NP que celui de l'évaluation. Il découvrit alors qu'un grand nombre de problèmes importants de recherche opérationnelle, en particulier celui du coloriage d'un graphe avec trois couleurs, sont aussi difficiles que tous les autres problèmes de la classe NP, c'est-à-dire sont NP-complets. On peut montrer directement, en transposant l'un des problèmes dans le contexte de l'autre, que le coloriage d'un graphe et l'évaluation sont équivalents.

On a ainsi pu démontrer, par des arguments analogues, que des centaines de problèmes que l'on croyait distincts, sont en fait équivalents et ne diffèrent que par leur notation. Tous ces problèmes sont équivalents au problème de l'évaluation et sont donc aussi NP-complets. On a aussi découvert d'autres ensembles de problèmes complets, tant pour la classe P que pour les classes de problèmes insolubles en pratique, c'est-à-dire ceux dont le temps de résolution à l'aide d'une machine de Turing croît exponentiellement avec la taille du problème. Il n'en demeure pas moins qu'une approche directe du problème P-NP semble prématurée. On peut se faire une idée de la difficulté de ce problème à travers les développements récents de l'informatique théorique.

A.3 La classe NP

NP signifie *Non deterministic Polynomial*. De façon intuitive on dit qu'un problème appartient à la classe NP si, disposant d'une solution (fournie par un devin), on peut vérifier en un temps polynomial que celle-ci résoud bien le problème.

De façon plus formelle, un problème appartient à la classe NP s'il peut être résolu en un temps polynomial par un algorithme non déterministe (un algorithme qui après chaque instruction à la possibilité de choisir la prochaine de façon optimale [2])

Par définition on ne connaît donc pas d'algorithme polynomial permettant de résoudre un problème de la classe NP, mais néanmoins il n'est pas possible d'affirmer qu'il n'en existe pas.

Par construction $P \subset NP$.

A.4 La classe NP-complet

Par définition on dit qu'un problème Q est dur pour une classe si, pour tout problème Q' de cette classe, il existe une transformation qui réduit toute instance de Q' en une instance de Q en un temps polynomial.

La classe des problèmes durs de NP est appelée « classe des problèmes NP-complets » (voir fig A.1). Cook en 1973 démontre que le problème de satisfaisabilité d'un ensemble de clauses de la logique propositionnelle est NP-complet.

L'étude des problèmes NP-complets est fondamentale en théorie de la complexité. En effet, si $P \neq NP$, les problèmes NP-complets ne peuvent être résolus exactement en un temps polynomial. Inversement, il suffit de montrer qu'un seul problème NP-complet peut être résolu en

[2] La notion d'algorithme non déterministe n'a pas de réalité physique. On peut néanmoins illustrer cette notion en supposant que l'on dispose d'une infinité de processeurs. Il est alors possible d'explorer en parallèle toutes les branches d'une alternative et donc d'effectuer pour chacune le choix optimal (*a posteriori*).

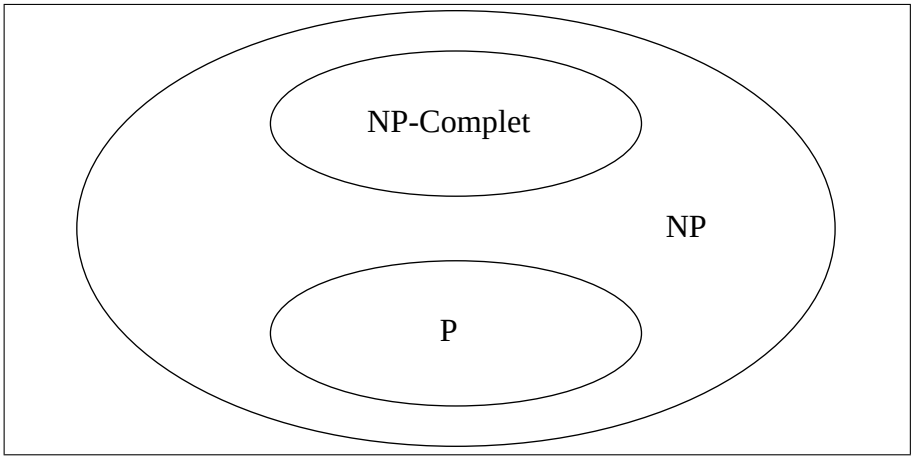

FIG. A.1 – Relations entre les diverses classes de problèmes.

un temps polynomial pour démontrer que tout problème NP-complet peut aussi être résolu en un temps polynomial ainsi que, bien sûr, tout problème NP. La conjoncture $P \neq NP$ semble cependant très probable.

Bibliographie