« Les mathématiques sont la science de l'infini, leur objet est la compréhension symbolique de l'infini avec l'humain, c'est-à-dire le fini »

Hermann Weyl

Peut-on trouver ce qu'on peut prouver ?

Même difficiles, certains problèmes ont la bonté de se laisser aborder : on peut y apporter une solution et prouver qu'elle est juste. D'autres sont moins conviviaux : il est seulement possible de prouver une réponse donnée. Mais prouver suffit-il pour trouver ? Répondre à cette question nous révélerait les limites universelles de ce que l'on peut faire par ordinateur.

András Sebö
est directeur de recherche au CNRS, au département de mathématiques discrètes du laboratoire Leibniz (IMAG), à Grenoble. Ses travaux portent en particulier sur l'optimisation combinatoire et la théorie des graphes.
www-leibniz.imag.fr/DMD/OPTICOMB/Membres/sebo/

Une réponse à la question posée par le titre du présent article coûterait un million de dollars au Clay Mathematics Institute, qui a lancé le concours *Millenium Prize Problems**. Sa difficulté, son importance pour les mathématiques, et plus encore ses conséquences en informatique rangent ce problème « P = NP? » (et d'autres de la même famille) parmi les plus grands défis actuels. Contentons-nous pour le moment de mentionner que les « cryptosystèmes » de l'informatique et des banques sont fondés sur l'hypothèse P ≠ NP. La publicité qui lui est faite a aussi conduit à des rumeurs et des idées fausses.

Comme beaucoup de problèmes difficiles de mathématiques qui deviennent les problèmes centraux de grandes théories, celui-ci a son intérêt scientifique et esthétique en soi. Mais ses applications augmentent considérablement l'enjeu d'une solution.

Comme la vitesse de la lumière en physique, il y a en informatique des limites qu'on ne pourra jamais dépasser

Il s'agit de comprendre les limites théoriques des ordinateurs, indépendamment du progrès technique. Comme la vitesse de la lumière en physique, il y a en informatique des limites qu'on ne pourra jamais dépasser. Ces limites viennent également de la réalité physique. Même pour des problèmes de taille raisonnable, le nombre d'opérations pour la résolution peut être déraisonnablement grand, rendant le problème intraitable indépendamment des moyens techniques. La lumière viendra donc probablement avant tout d'avancées d'ordre général dans la théorie de la complexité. L'interrogation de base porte sur les limites de ce qu'on pourra *trouver un jour*.

La Nature – le « suprême fasciste » comme dirait Paul Erdös[1] – est particulièrement diabolique ici : elle met une grande partie des problèmes appliqués dans la famille des problèmes « universels » ; ces problèmes pour lesquels une hypothétique « formule miracle » qui résoudrait l'un d'eux permettrait en réalité de les résoudre tous. Pour démontrer qu'un problème est intraitable, la principale manière de faire est aujourd'hui de montrer qu'il est universel[2,3].

Avant d'aller trop loin, plongeons-nous dans des maths simples. Un nombre est appelé *composé* s'il s'écrit comme le produit de deux nombres entiers plus grands que 1. Un nombre est appelé *premier* s'il n'est pas composé. Par exemple 6 = 2 × 3 est composé ; 7 est premier.

A votre avis, le nombre suivant, fait de 165 chiffres, est-il composé ? Ou est-il premier ?

5506052109362772219442854813805821351374043788221029083366595041386180507252593063230692920076595459969368537919645619024071175249445395206547458950792107906048129 03

Donner une preuve. Répondre « oui » à la première question, c'est répondre « non » à la deuxième. Répondre « non » à la première, c'est répondre « oui » à la deuxième. Le problème est de convaincre son interlocuteur que la réponse est la bonne. Il faut lui en donner la preuve. Il faut donc qu'une preuve courte existe. Pour vous *convaincre* qu'un nombre est composé, il existe au moins une preuve : sa présentation comme produit de deux nombres. Pour vérifier que le nombre est premier, sans avoir à en chercher le produit, les mathématiciens ont une recette (un *algorithme*) réalisable (*polynomial*) – même si pour le moment celle-ci fait appel soit à l'hypothèse de Riemann (voir l'article de Gilles Lachaud), soit au hasard (algorithme aléatoire)[4]. Autre bonne nouvelle, du moins pour la cryptographie : personne ne saurait actuellement *trouver* deux nombres entiers dont ce nombre est le produit, même les experts les plus brillants, armés des plus puissants ordinateurs du monde[1]. Personne, sauf vous et moi. Voilà :

5506052109362772219442854813805821351374043788221029083366595041386180507252593063230692920076595459969368537919645619024071175249445395206547458950792107906048129 03 =

19495170322564563517577564636611437406357821662441854661255847397144899709039156859112208250489021079126258105 7 × 282431598096366795258333674158635131923014625328011429

(1) Schechter, *My Brain is Open*, Simon & Schuster, 1998.

(2) Garey et Johnson, *Computers and Intractability*, Freeman, 1979 ; *NP completeness Column : an Ongoing Guide*, Journal of Algorithms.

(3) Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

(4) Rabin, *Probabilistic Algorithms*, in *Algorithms and Complexity*, Academic Press, 1976.

En deux mots

A défaut de pouvoir résoudre tous les problèmes, les mathématiciens cherchent à savoir quels problèmes leur sont accessibles, c'est-à-dire pour lesquels ils pourront concevoir un algorithme qui aboutit en un temps raisonnable (ou polynomial). De même, ils cherchent à identifier les problèmes pour lesquels, indépendamment des progrès techniques, il ne leur sera jamais possible d'apporter une réponse. C'est peut-être le cas des problèmes *universels* (ou NP-complets) : si l'on parvenait à en résoudre un, alors tous seraient résolus. On aura alors démontré que P = NP. Mais il paraît peu probable que l'on puisse un jour concevoir un algorithme polynomial pour les problèmes universels même si, pour le démontrer, des idées radicalement nouvelles devront probablement être apportées à la théorie de la complexité.

La Recherche a publié :

(I) Henri Cohen, « Cryptographie à clé publique et nombres premiers », dans « L'intrigue des nombres premiers », hors-série n° 2, août 1999.

(II) Jean-Michel Kantor, « Leonhard Euler » ; Benoît Rittaud « Des sommets et des arêtes pour retrouver son chemin », numéro exceptionnel *Pour mathématiques*, mai 2000.

J'avoue : j'ai triché. J'ai d'abord pris deux nombres, puis je les ai multipliés. C'est seulement par paresse que j'ai utilisé un ordinateur : sur une très, très grande feuille, on pourrait faire la multiplication à la main. Si vous refaites cela, vous aurez vous-même *prouvé* que le nombre donné était composé ! Vous en tenez le témoignage irréfutable : la décomposition en produit. Prouver est synonyme ici de « (se) convaincre », « vérifier », « certifier ». Pour la preuve des théorèmes, nous utilisons plutôt le terme « démonstration ».

On peut donc prouver sans trouver. Mais peut-on prouver qu'on ne trouvera pas ? Comment prouver que demain l'on ne saura pas trouver rapidement le produit dont est fait n'importe quel nombre composé ?

Afin de prouver qu'il allait trouver les Indes, Christophe Colomb a probablement « arrangé » les données de son projet et réduit la taille supposée du globe terrestre : la circonférence était en effet trop grande pour qu'il puisse traverser l'océan dans le temps imparti.
© Bridgeman-Giraudon

Au-delà de la cryptographie, quel est l'intérêt de ce genre de question ? C'est que si on pouvait trouver tout ce qu'on peut prouver, une foule de problèmes réputés insolubles seraient résolus.

Il n'est pas très réaliste de croire que l'on pourra trouver tout ce que l'on peut prouver, mais les mathématiciens veulent au moins cerner ce que l'on a des chances de trouver, et identifier ce que l'on n'a aucune chance de trouver. Il est raisonnable de prévoir qu'il sera possible de démontrer que certains types de problèmes ne pourront jamais être résolus en un temps réalisable.

Supposons qu'à la suite d'un vol un inspecteur de police, après de longs mois de travail, trouve un suspect. Il doit maintenant prouver qu'il s'agit bien du coupable. Les juges veulent la preuve, le témoignage irrécusable : des empreintes digitales, etc. Si le policier se trompe, l'accusé a intérêt à *prouver* son innocence : il doit produire son propre témoignage irrécusable, par exemple un alibi. En revanche, il ne pourra pas prouver son innocence grâce à l'absence d'empreintes. A la question : l'accusé est-il innocent ? l'alibi et les empreintes délivrent le certificat que la réponse est oui ou non. Une fois le certificat obtenu, la réponse est facile. Mais comment trouver le certificat ?

Dans notre vie, les certificats sont chose courante. Ils visent à remplacer une procédure laborieuse par un document court. Plus ils sont difficiles à trouver, plus ils sont précieux. N'est-il pas plus simple de demander un permis de conduire que de vérifier l'aptitude de quelqu'un à conduire ? De regarder un carnet de santé, que de partir à la quête des vaccins et des maladies passées ?

Autre exemple, les recettes de cuisine : ce sont de véritables *algorithmes* : des procédures générales bien définies. Avec des *données* précises (les ingrédients), elles nous donnent des instructions précises qui conduisent à un *résultat* précis.

Certaines recettes sont longues à réaliser. Vérifier la qualité du plat tout prêt est beaucoup plus simple. Vérifier (certifier, prouver) qu'un certain résultat est juste et faire le chemin jusqu'au résultat (trouver) ont tous deux leur importance, mais ce ne sont pas des procédures de même nature et elles n'ont pas la même fonction : nous avons souvent besoin de montrer le résultat de notre travail, sans faire perdre de temps à ceux qui nous écoutent…

Preuve et existence. La différence entre *pouvoir prouver* quelque chose en un temps réalisable, et *trouver* ce quelque chose – même s'il n'y a que deux possibilités, comme *composé* ou *premier*, *coupable* ou *non coupable*, *réussi* ou *raté* – est souvent significatif : pour prouver, il peut suffire de vérifier les empreintes ou l'alibi ; décider ou trouver, c'est toute l'enquête. Ce sont ces deux notions et leurs relations qui sont les clés de l'analyse de la complexité des algorithmes.

L'existence d'un certificat, d'un témoignage irrécusable à une réponse « oui » pourrait-elle impliquer qu'il est possible de « décider » *a priori* si la réponse est « oui », et aussi de trouver les certificats ? Il doit bien exister une corrélation entre l'existence d'une preuve courte, et une procédure réalisable pour trouver la preuve ! Mais on conçoit aussi qu'il soit plus difficile de fabriquer quelque chose que de l'apprécier. L'art est plus difficile que la critique !

C'est la question centrale : savoir si la solution des problèmes que l'on rencontre est ou non accessible. Si l'on sait que la solution est accessible, on peut s'atteler à la tâche et la trouver. Si l'on sait qu'elle n'est pas accessible, inutile de perdre son temps.

Les problèmes de P (problèmes que l'on peut résoudre en un temps *polynomial*) sont les problèmes de décision pour lesquels la réponse « oui » ou « non » peut être trouvée (décidée) en un temps réalisable. Pour la classe des problèmes NP (non déterministiquement polynomial) en revanche, on sait seulement que la réponse « oui » à la question posée peut être vérifiée (certifiée, prouvée) en un temps réalisable ; on peut convaincre un tiers de la réponse « oui » si elle est vraie, mais on ne peut pas nécessairement trouver cette réponse.

*Un siècle après les 23 problèmes que David Hilbert considérait comme essentiels à résoudre, le concours **Millenium Prize Problems**, lancé l'an passé par la fondation Clay, porte sur sept grands problèmes qui ont tous résisté aux efforts des chercheurs pendant des décennies. Un million de dollars sera offert pour la résolution de chacun d'eux.
www.claymath.org/prizeproblems/index.htm

La présence d'empreintes digitales peut permettre de confondre un suspect en apportant la preuve de sa culpabilité. L'absence d'empreintes en revanche ne le disculpe pas. © BSIP Estiot

Le grand problème que les mathématiciens ne savent toujours pas résoudre est de savoir si l'ensemble P des problèmes est le même que l'ensemble NP. On écrit : « P = NP ? ». C'est-à-dire : peut-on trouver tout ce qu'on peut prouver (par exemple, les deux nombres entiers dont un nombre est le produit) ? L'inverse est vrai, bien sûr, c'est-à-dire que P est entièrement inclus dans NP. S'il existe un moyen de trouver rapidement, la personne n'a qu'à exécuter l'algorithme pour se convaincre. L'algorithme prend alors lui-même valeur de certification.

La distinction entre l'existence d'objets mathématiques et leur construction est relativement récente. Le souci de prendre en considération le temps d'exécution des algorithmes n'apparaît que dans les années 1950-1960, dans plusieurs travaux, dont ceux de John von Neumann bien après les travaux d'Alan Turing, le concepteur du premier modèle mathématique de l'ordinateur. Quant aux notions de P et de NP, elles ont été suggérées en 1965 dans un article de Jack Edmonds[5].

Dans la lignée de Gödel. Le problème mathématique consiste donc à démontrer que deux ensembles sont égaux. Cependant, la solution du problème demande probablement plus qu'un grand talent de mathématicien. Comme il y a cent ans avec l'axiomatisation de la théorie des ensembles, il est possible que les notions et les méthodes avec lesquelles nous approchons le problème soient elles-mêmes inadaptées. Le million de dollars n'est pas facile à gagner ! Notre problème s'inscrit dans la lignée des résultats de Gödel sur l'indécidabilité de certaines propositions, mais est probablement beaucoup plus difficile.

Redescendons sur terre. Il y a beaucoup de problèmes pour lesquels il est évident qu'ils sont dans P. Par exemple, décider si oui ou non un nombre est divisible par 2, 3, ou par n'importe quel nombre. Un problème classique, un peu moins évident mais encore facile : étant donné le plan d'une ville, vérifier si l'administration urbaine a bien fait son travail et qu'on peut aller de n'importe quel endroit à n'importe quel autre, en respectant les rues en sens unique.

En voici un autre : le problème du *club à deux salles*. Dans un club, on sait exactement qui connaît qui. On a deux salles pour organiser un pot auquel tous les membres du club sont conviés, et l'on cherche à éviter qu'il y ait deux personnes dans la même salle qui se connaissent déjà. Peut-on décider à quelles conditions c'est possible ?

Vous avez certainement trouvé : il *suffit* qu'il y ait trois personnes du club qui se connaissent mutuellement pour ne plus pouvoir placer les gens. Cette condition est *suffisante*, mais est-elle *nécessaire* ?

Observons ce qu'il se passe avec 7 personnes, placées initialement autour d'une table ronde, chacune d'elles connaissant uniquement ses deux voisins. Peut-on les séparer afin de les placer dans les deux salles, comme désiré ? Si l'on répartit les personnes successivement, dans l'ordre de leur placement autour de la table, on est forcé d'alterner entre les deux salles. Tout va bien jusqu'à la sixième personne, mais nous sommes dans l'incapacité d'orienter la septième : elle connaît déjà une personne dans chaque salle. Ce cas de figure doit être aussi exclu. Nous avons donc avancé par rapport à notre première idée, mais de nouveau cette condition peut ne pas être *nécessaire* : il pourrait y avoir d'autres obstacles. En fait, il n'y en a pas. La démonstration est relativement facile : on peut procéder comme nous venons de le faire pour sept personnes, en plaçant les gens dans deux salles, et quand ce n'est plus possible autour d'une table avec un nombre impair de places de façon à ce que chacun connaisse ses deux voisins (pour la procédure exacte, qui est un exemple simple de démonstration par algorithme, voir www.larecherche.fr). D'où la conclusion : *on peut mettre tous les membres du club dans deux salles, si et seulement si on ne peut pas sélectionner des personnes du club et les placer autour d'une table ronde avec un nombre impair de places de façon à ce que chacun connaisse ses deux voisins.*

L'essence du problème. Il existe donc une procédure de validation de la question : un placement en deux salles est-il possible ou non ? L'existence de cet algorithme fait que le problème est dans P, et le théorème que nous avons formulé le met déjà dans NP.

Bien que l'algorithme qui permet de le démontrer soit simple, le théorème contient l'« essence » du problème. Il permet d'obtenir une solution avec environ $c \times 1\,000 \times 1\,000$ opérations, où c est une petite constante – un ordinateur moyen fait ce calcul en quelques secondes. Autrement, nous pourrions être amenés à tester une par une les 2^n possibilités de placer n personnes dans deux salles. Pour mille personnes, ceci ferait déjà 2^{1000} possibilités.

Décider des sens uniques dans une ville est un problème complexe. Mais vérifier que l'administration urbaine a bien fait son travail est facile : on peut trouver rapidement un chemin permettant d'aller à n'importe quel endroit. Le problème est « dans P ». © Israël Talby/Rapho

(5) Edmonds, *Canadian J. of Mathematics*, 17, 3, 449, 1965

(6) Berge, *Sur une conjecture relative aux codes optimaux*, Comm. 13e assemblée générale de l'URSI, Tokyo, 1962.

(7) Berge, *Graphes et Hypergraphes*, Dunod, 1970.

(8) Ramirez-Alfonsin, Reed, *Perfect Graphs*, collection d'articles, Springer, à paraître.

Lorsque le but est de trouver le plus court chemin en empruntant toutes les routes, le problème du « voyageur de commerce » s'appelle le problème du « postier chinois ». Un nom que Jack Edmonds a choisi en l'honneur d'un professeur chinois, qui fut le premier à traiter le problème dans les années 1960.

© *Jour de Fête*, Jacques Tati, cahiers du cinéma

Le mathématicien irlandais William Rowan Hamilton (1805-1865) nous a laissé d'importants travaux en optique, en dynamique et en mathématiques. Il fut notamment le fondateur de la théorie des nombres complexes. © Harlingue-Viollet

Le théorème nous permet donc de ramener le nombre exponentiel de cas à un nombre plus petit réalisable

Et il y a à peu près le même nombre de placements autour de tables impaires. Aucun ordinateur ne sera jamais capable de traiter tous les cas de figure. Le théorème nous permet donc de ramener le nombre exponentiel de cas à un nombre plus petit, réalisable.

Mais pour certains problèmes, personne n'arrive à donner une solution réalisable en un temps accessible à l'homme et aux ordinateurs ou à démontrer un théorème qui saisit l'« essence » du problème. C'est déjà le cas pour les clubs à trois salles.

Vous pouvez commencer par généraliser la méthode que nous avons employée avec deux salles et trouver des obstacles (par exemple des quadruplets de personnes qui se connaissent). Mais votre liste de certificats courts ne s'arrêtera jamais sans qu'il y ait un contre-exemple. Ou plutôt, si elle s'arrête, vous aurez résolu l'un des grands problèmes de l'informatique, car le problème des trois salles est « universel ».

Pour d'autres problèmes encore, on ne sait pas s'ils sont universels ou s'ils admettent une solution réalisable. C'est le cas du test de primalité d'un nombre. C'est le cas également de la conjecture des graphes parfaits que le mathématicien français Claude Berge – le doyen de la *Théorie des graphes*[6] – a énoncé en 1960[7]. Cette conjecture et les problèmes algorithmiques qui lui sont liés ne sont toujours pas résolus. Mais les mathématiciens n'ont pas renoncé, comme en témoigne la réunion d'une trentaine d'entre eux autour de ces problèmes durant une semaine en septembre dernier à Princeton. C'est que les conséquences théoriques et appliquées d'une démonstration seraient importantes[8] (plus de précisions sur www.larecherche.fr).

Voyons un autre problème, celui du *cycle hamiltonien* : étant donné un ensemble de villages avec la carte des routes qui les relient, peut-on faire une tournée en passant par chaque village exactement une fois, et en revenant au village initial? Une tournée qui passe par chaque village exactement une fois s'appelle cycle hamiltonien, du nom du mathématicien irlandais William Rowan Hamilton, inventeur du « jeu du dodécaèdre » (un cas particulier du célèbre problème du « voyageur de commerce »). La délivrance d'un certificat pour le « oui » est évident : il suffit de montrer une tournée. En revanche, on ne sait pas délivrer de certificat pour le « non ». Certes, lorsque le nombre de villages est petit, il est possible de trouver une preuve courte, mais personne ne sait démontrer qu'en augmentant la taille du problème, celle des preuves ne va pas augmenter exponentiellement.

Cycles hamiltonien et eulérien. Ce problème ressemble beaucoup à celui du tour eulérien : étant donné un ensemble de villages avec la carte des routes qui les relient, peut-on faire une tournée en passant par chaque route exactement une fois, et en revenant au village initial ? Une tournée qui passe par chaque route exactement une fois s'appelle « tour eulérien »[II]. La seule différence par rapport au problème précédent est que nous avons remplacé « village » par « route ». La nuance n'est pas mince : le problème du cycle eulérien, lui, est dans P. La solution est donnée par le théorème d'Euler, qui ressemble à celui formulé pour le club à deux salles : *il existe un cycle eulérien si et seulement si on peut aller de n'importe quel village à n'importe quel autre, et que le nombre de routes qui partent de chaque village est pair.*

Pour les mathématiciens qui les développent, les algorithmes réalisables (ou polynomiaux) sont ceux dont le comportement pour des données (des occurrences) arbitrairement grandes se décrit de la façon suivante : pour tout n, pour des données qui n'occupent pas plus de n octets sur un disque dur, l'algorithme s'exécute en moins de Kn^k opérations élémentaires, où K et k sont des constantes (les unités importent peu : cette définition impose seulement la manière dont le temps d'exécution augmente en fonction des données). Le terme *algorithme polynomial* vient du fait que la fonction Kn^k

est un polynôme. L'opposition entre « exponentiel » et « polynomial » est en quelque sorte du même ordre que celle entre le nombre et le nombre de chiffres qui le composent.

Il est remarquable que des problèmes aussi différents puissent être réduits les uns aux autres

Récapitulons. P est l'ensemble des problèmes de décision qu'on peut résoudre avec un algorithme polynomial. NP est l'ensemble des problèmes de décision pour lesquels il existe un certificat de longueur polynomiale pour la réponse « oui ». L'ensemble des problèmes pour lesquels il existe un certificat de longueur polynomiale pour la réponse « non » s'appelle « coNP ». P est inclus dans NP, ainsi que dans coNP. Finalement, nous avons toujours posé dans nos exemples les trois mêmes questions :

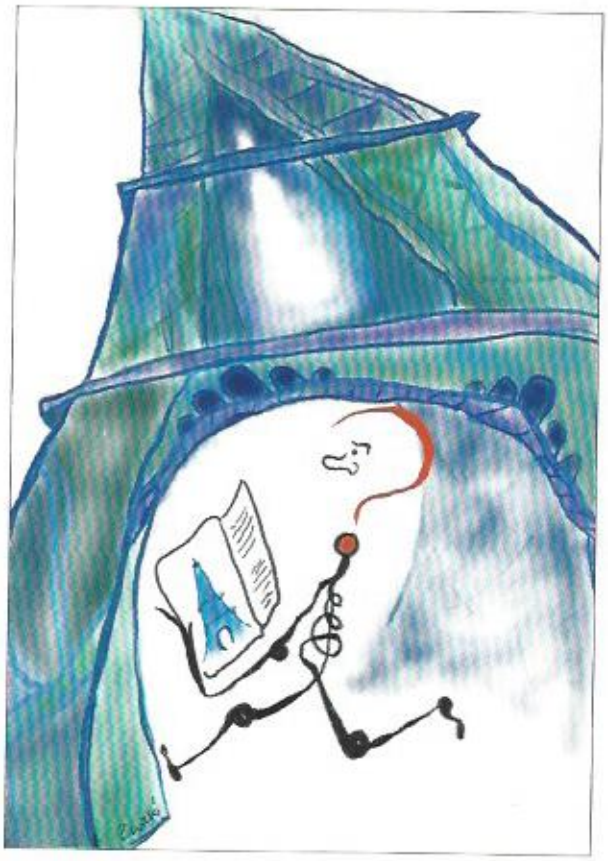
1) Le problème est-il dans NP et est-il dans coNP? (La réponse « oui » ou « non » peut-elle être *certifiée*, prouvée avec une vérification réalisable?)

2) Le problème est-il dans P? (La réponse « oui » ou « non » peut-elle être *trouvée, décidée* en un temps réalisable?)

3) Un certain certificat du « oui » ou du « non » peut-il être *trouvé* en un temps polynomial?

Jusqu'à présent nous avons négligé la différence entre la deuxième et la troisième question : trouver la réponse « oui » ou « non », ou un certificat du « oui » ou « non », nous a été égal. Si P = NP, alors, pour un problème donné, la réponse à l'une des interrogations de la première question est « oui », si et seulement si, il en est de même pour toutes les autres questions.

Dans le problème du club à deux salles, nous avons vu que la réponse à toutes les questions est « oui ». Le problème du cycle hamiltonien est dans NP, mais on ne sait pas s'il est dans coNP. On ne sait même pas si la réponse à la deuxième question est positive ou négative, donc la troisième question ne se pose même pas. Comme tous les problèmes de décision *universels* ou *complets* dans NP (*NP-complets*), le problème du cycle hamiltonien est un problème de NP auquel *tous* les problèmes de NP peuvent être *réduits*. Autrement dit : un problème est NP-complet si, en le résolvant, on peut résoudre tous les problèmes dans NP. Il est remarquable que des problèmes aussi différents que ceux qui relèvent des mathématiques discrètes ou de la théorie des nombres puissent être réduits les uns aux autres.



« Je suis sûr qu'elle existe, mais où est-elle ? »
Pour certains problèmes, trouver peut être bien plus difficile que prouver.
Illustration Suzanne Sebá

En 1971, Stephen Cook[9] et indépendamment Leonid Levin ont présenté des problèmes concrets auxquels pouvaient être réduits tous les problèmes de NP en un temps polynomial. Il a ainsi démontré le théorème fondamental suivant : *il existe un problème universel dans NP.*

Peu après, le problème de Cook a lui-même été réduit à beaucoup de problèmes importants, dont celui du cycle hamiltonien, ou à celui du club à trois salles (que l'on appelle également de « 3-coloration des graphes »). Depuis, la liste des problèmes NP-complets grandit vite (plus de mille sont connus à ce jour et les démonstrations de NP-complétitude sont devenues pratique courante). Il est probable que votre jeu préféré soit un problème NP-complet, comme c'est le cas du Sokoban, du solitaire ou du démineur.

Informaticiens, mathématiciens et ingénieurs utilisent la liste des problèmes NP-complets comme un traducteur utilise son dictionnaire. Savoir qu'un problème est NP-complet signifie avant tout que le problème est *trop* difficile. Pour les experts, la conclusion alors est de *renoncer à travailler sur un algorithme réalisable* pour un tel problème dans sa forme originale. La théorie de la complexité sert en quelque sorte à ne pas perdre son temps quand les choses ne fonctionnent pas ou à être réaliste dans ses attentes.

Il existe par ailleurs des problèmes pour lesquels on ne sait pas démontrer leur appartenance à NP, mais dont on sait qu'ils sont *NP difficiles*, c'est-à-dire que si l'on connaissait un algorithme qui les résout, on pourrait venir à bout de tous les problèmes de NP. Notons qu'il existe également d'autres classes de questions, comme « NP ∩ coNP = P? » où « NP ∩ coNP » est l'ensemble des problèmes (dits *bien caractérisés*) qui appartiennent à la fois à NP et à coNP, c'est-à-dire pour lesquels la réponse « oui » ainsi que la réponse « non » peuvent être prouvées (comme pour le club à deux salles ou le test de primalité).

Le non-dit. Si quelques mathématiciens réfléchissent sur les problèmes les plus théoriques de l'informatique, un grand nombre de chercheurs travaillent avant tout sur des problèmes concrets : des théorèmes de bonne caractérisation, des algorithmes polynomiaux, ou, faute de mieux, sur des démonstrations de NP-complétitude. Peut-être ne pensent-ils que rarement à la question : P = NP ? On pense généralement – sans le dire – que P ≠ NP. Quand un problème est NP-complet, on se console avec des algorithmes d'approximation ou d'autres compromis (comme les algorithmes « évolutifs » qui s'inspirent des processus biologiques ou physiques), ou d'autres modèles de calculs (comme la complexité quantique). Dans chaque situation, certaines méthodes se révèlent plus adaptées que les autres : il n'y a pas de solution miracle valable pour tous les problèmes.

Parmi les trois questions que nous avons posées précédemment, la troisième est peut-être la plus intéressante : est-ce qu'un certain certificat du

[9] Cook, *Conference records of 3d annual ACM Symposium on Theory of Computing*, 151, 1971.

« oui » ou du « non » peut être *trouvé* en un temps polynomial? Sa distinction des problèmes de décision est souvent mal ou peu connue, car elle relève de résultats accumulés au cours de ces dix dernières années et qui ne sont pas tous assimilés.

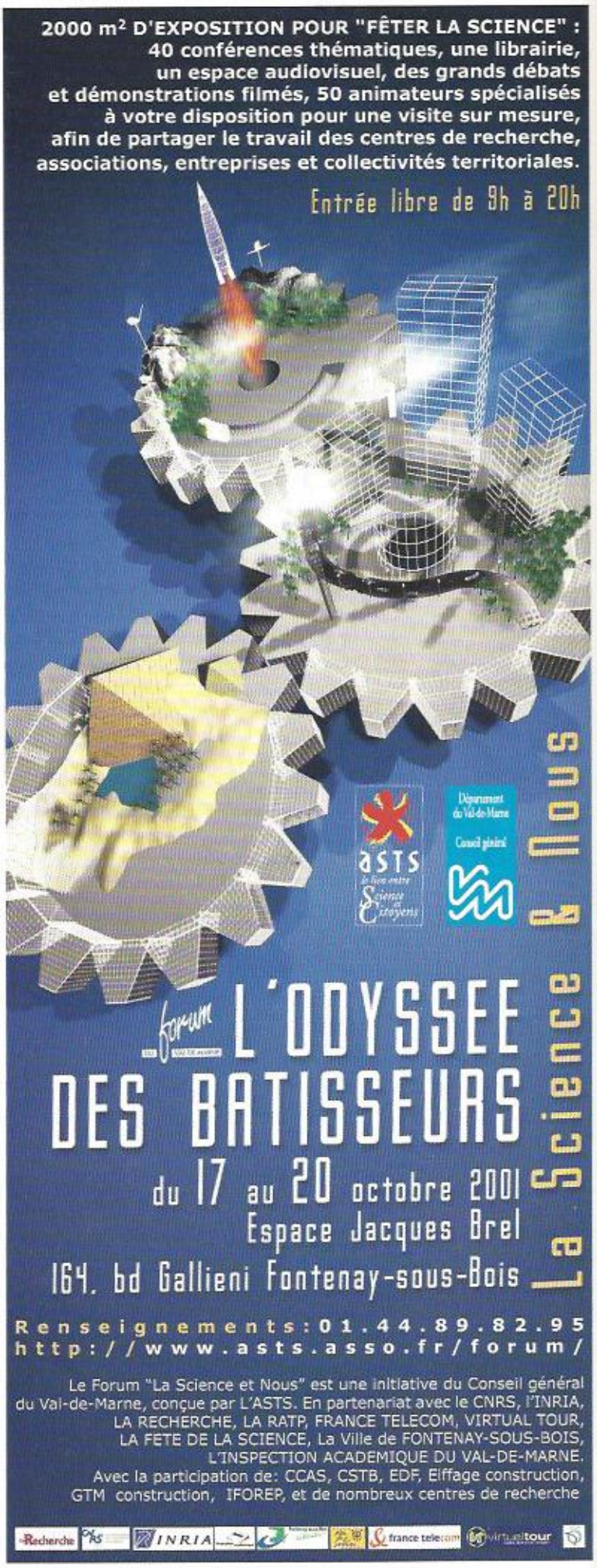
Il est clair que trouver n'est pas moins difficile que décider, et décider n'est pas moins difficile que certifier une décision. Si on sait trouver, on sait décider; si on sait décider, on sait prouver. Quelle est la relation exacte entre *trouver*, *décider* et *prouver*? La question P = NP telle qu'on l'a énoncée signifie : peut-on décider (trouver si la réponse est « oui » ou « non ») si on peut prouver la réponse « oui »?

Une erreur très commune est de penser que *décider* et *trouver* sont à peu près équivalents. Pourtant, la différence existe, et elle est souvent d'autant plus grande que le problème est facile. Par expérience on sait seulement que, pour tous les problèmes NP-complets, *décider* et *trouver* sont soit tous les deux réalisables, soit aucun ne l'est.

Et si P = NP ? Beaucoup de mathématiciens qui ont pris part au début de la théorie de la complexité pensent toujours que, pour les problèmes bien caractérisés, trouver le certificat ne peut être difficile. Cela n'a en fait rien d'évident. Il a été montré que si P ≠ NP, alors il existe des problèmes qui ne sont ni dans P ni NP-complets, ainsi que des problèmes pour lesquels *trouver* est plus difficile que *prouver*. Posons le problème de la factorisation un peu différemment : « Un nombre donné admet-il une factorisation en nombres premiers ? » Chacun sait que la réponse est toujours oui, donc le problème de décision est évidemment dans P. On peut certifier une telle factorisation en un temps polynomial en donnant un certificat de primalité pour chaque nombre premier qui y participe (plus de précisions sur www.larecherche.fr). Mais, comme nous l'avons vu, personne ne sait la trouver !

Si par hasard il apparaissait que P = NP alors toutes les classes de complexité que nous avons définies, et bien d'autres, deviennent égales. Tous les problèmes de P deviennent alors NP-complets, et il n'y a plus de différence entre la complexité de prouver, de décider et de trouver. Inversement, bien sûr, si on sait trouver ce qu'on sait prouver, alors P = NP. Certes, vous pourrez estimer que vous n'avez pas avancé par rapport au début de l'article. On voit seulement peut-être mieux ce qu'on ne connaît pas, et, avec Socrate, penser que c'est un progrès. Pendant ce temps, le « suprême fasciste » met les problèmes qui nous intéressent parmi les problèmes universels qu'on ne réussit pas à résoudre. Et il le fait plus souvent qu'on ne le souhaiterait…

A.S. ■

A lire :

- Lewis Papadimitriou, « L'efficacité des algorithmes », dans *L'Intelligence de l'informatique*, Bibliothèque pour la science, 169-182, 1988.
- Wilfe, *Algorithmes et complexité*, série « *Logique mathématiques informatique* », Masson, 1989.
- Cipra, *What's New in the Mathematical Sciences*, série de l'AMS.