Peut-on trouver ce qu'on peut prouver ?

Des mathématiciens à la recherche de problèmes « universels »

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.

Illustrations
Laurent Taudin

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 à 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.

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].

Jusqu'à l'apparition des demandes de l'informatique, on pouvait peut-être dire avec Hermann Weyl, que *« 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. »*

Aujourd'hui, le « fini, mais très grand » y a pris sa part. En effet, réduire le très grand en réalisable, c'est rendre possible ce qui était impossible. On pourrait donc réécrire la citation de Weyl : *« Un des objets des mathématiques, c'est la réduction de l'intraitablement grand à l'échelle de notre monde physique, c'est-à-dire au traitable. »*

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 x 3 est composé ; 7 est premier.

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

5506052209562772289442854815805821351374045788221
0290856695045866805072525930632306929200765954599695685537919645619024071175249445395206547458950792079060481290 5

Convaincre son interlocuteur. 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 convaincre que le nombre est premier, il existe aussi des preuves simples que l'on peut expliquer à des lycéens. De plus, on peut même trouver la réponse à la question « ce nombre est-il ou non premier ? » avec une recette (algorithme) réalisable (polynomial) – même si pour le moment celle-ci fait appel soit à l'hypothèse de Riemann, soit au hasard (algorithme aléatoire)[4]. Autre bonne nouvelle, du moins pour la cryptographie : même si la réponse trouvée est « non », personne ne saurait actuellement trouver deux nombres plus grands que 1 dont

*Un siècle après les 23 problèmes que David Hilbert considérait comme essentiels à résoudre, le concours **Millenium prize problems**, lancé 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

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

(2) M.R. Garey, D.S. Johnson, *Computers and Intractability*, Freeman, 1979 ; « NP-completeness column: an ongoing guides », *Journal of Algorithms*.

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

(4) Rabin, « Probabilistic algorithms », *Algorithms and Complexity*, Academic Press, 1976.

Cet article est la version revue et mise à jour de « Peut-on trouver ce qu'on peut prouver ? », *La Recherche* n° 346, pp. 64-69.

La Recherche a publié :

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

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

le nombre est le produit, même les experts les plus brillants, armés des plus puissants ordinateurs du monde[1]. Personne, sauf vous et moi. Voilà :

5506052109362772219442854813805821351374043788221
0290835665950413861803072525950632306929200765954599695685379196456190240711752494453952065474589307921079060481290 3 =
1949517032256456351757756465661145740635782166244185466123584739714489970903915685911220823048902107912623810 57 x 2824315980963667952585356741586351319230146255280114 29

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 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 ?

Pas très réaliste. Au-delà de la cryptographie, quel est l'intérêt de ce genre de question ? C'est que si l'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?

Recettes. 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...

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.

Pour prouver, il peut suffire de vérifier les empreintes ou l'alibi ; décider ou trouver, c'est toute l'enquête

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, et si quelqu'un nous souffle la preuve, mais on ne peut pas nécessairement la trouver.

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].

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.

Sens unique. 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 sept personnes, placées initialement autour d'une table ronde, chacune

[5] J. Edmonds, *Canadian J. of Mathematics*, 17, 3, 449, 1965.

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 pourrions être amenés à tester une par une les 2^{1000} possibilités de placer 1 000 personnes dans deux salles

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.

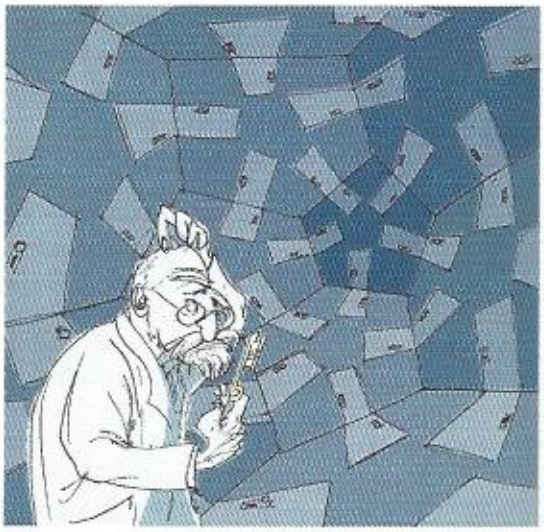
Il existe donc une procédure de validation de la question: un placement en deux salles est-il possible? Impossible ? L'existence de cet algorithme fait que le problème est dans P, et le théorème que nous avons formulé met déjà dans NP les deux questions. Pour 1 000 personnes, même si l'on ne fait aucun effort supplémentaire, l'algorithme va conduire en $c \times 1\,000 \times 1\,000$ 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^{1000} possibilités de placer 1 000 personnes dans deux salles. Pour mille personnes, cela ferait déjà 2^{1000} possibilités. 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! L'algorithme arrive donc à ramener le nombre exponentiel de cas à un nombre plus petit, réalisable. Cela est possible seulement si l'essence du problème est saisie!

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 du problème qui consiste à tester si un graphe* est parfait. Ce problème a été promu par la conjecture des graphes parfaits que le mathématicien français Claude Berge – un des grands pionniers de la théorie des graphes[6] – a énoncé en 1960[7]. Cette conjecture et les problèmes algorithmiques qui lui sont liés viennent d'être résolus, comme cela a été annoncé le 23 mai 2002 (voir l'encadré : «Les quarante-deux ans de la conjecture de Berge», p. 25).

Cycle hamiltonien. Afin d'éclaircir les notions que nous avons introduites, passons à 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 évidente: 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 de la non-existence d'un tour ne va pas augmenter exponentiellement.



Un autre problème qui semble *a priori* peu différent: 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»[11]. La seule différence par rapport au problème précédent est que nous avons remplacé «village» par «route». La nuance n'est pas mince: le

*Un **graphe** est un ensemble de paires d'éléments choisis dans un ensemble donné.

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

(7) C. Berge, *Graphes et hypergraphes*, Dunod, 1970.

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. 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 du même ordre que celle entre le nombre et le nombre de chiffres qui le composent.

Informaticiens et mathématiciens utilisent la liste des problèmes NP-complets comme un traducteur son dictionnaire

Trois questions. 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 ? Quand les deux peuvent, on a de bonnes chances que le problème soit aussi dans P.)
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.

Votre jeu préféré. En 1971, Stephen Cook[8] 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 polynomial 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 « $NP \cap coNP = P$? » où « $NP \cap coNP$ » 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).

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

*Un graphe est dit **parfait** si, en supprimant un ensemble quelconque de sommets, la clique maximum (« le nombre maximum de membres du club qui se connaissent mutuellement ») est égale au nombre chromatique (« le nombre de salles minimum nécessaires pour placer ces membres » sous les contraintes expliquées dans l'article).

*Un **trou impair** est un cycle impair sans corde, c'est-à-dire un ensemble de « membres d'un club que l'on peut asseoir autour d'une table ronde », de taille impaire supérieure ou égal à 5 de façon à ce que les voisins et seulement les voisins se connaissent.
Un **antitrou impair** est le complémentaire d'un trou impair, c'est-à-dire un ensemble de membres que l'on peut asseoir autour d'une table ronde, de taille impaire supérieure ou égale à 5 de façon à ce que les voisins et seulement les voisins ne se connaissent pas.
Il est facile de voir que les trous ou les antitrous ne sont pas parfaits, et donc les graphes qui les contiennent ne le sont pas non plus.

[8] S. Cook, Conference records of 3rd Annual ACM Symposium on Theory of Computing, 151, 1971.

[9] J. Ramirez-Alfonsin (ed.), Perfect Graphs, Ed. Wiley

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.

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

Beaucoup de mathématiciens qui ont pris part au début de la théorie de la complexité pensent toujours que, pour les problèmes bien caractérisés (qui sont dans NP et aussi dans coNP), 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 trouver le certificat de non-primalité qui consiste à écrire un nombre comme un produit.

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 on découvre que le « suprême fasciste » met un grand nombre de problèmes qui nous intéressent parmi les problèmes universels. Et il le fait plus souvent qu'on ne le souhaiterait… **A.S.** ■

Les quarante-deux ans de la conjecture de Berge

Un graphe est parfait si et seulement si il ne contient pas de trou ni d'antitrou impairs**. Cette conjecture, appelée aussi « conjecture forte des graphes parfaits » a été considérée comme l'un des plus grands défis de la théorie des graphes. Elle a été énoncée en 1960 par Claude Berge, motivé entre autres par la théorie des jeux et par des problèmes de la théorie de l'information (surtout la capacité de Shannon). Au cours du temps, les graphes parfaits et les idées qu'ils ont stimulées ont pris de l'importance bien au-delà de la théorie des graphes : par exemple, dans la théorie de la combinatoire polyédrale, ou récemment dans l'application des méthodes de la programmation semi-définie en mathématiques discrètes, mais on trouve aussi des applications dans la conception de grands circuits intégrés.

Pendant près de quarante-deux ans, cette conjecture a retenu l'attention d'un grand nombre de chercheurs, dont celle de plusieurs équipes françaises ; des centaines d'articles lui ont été consacrés… Elle a passé sa première douzaine d'années en compagnie de sa sœur cadette, la conjecture faible, démontrée en 1972 par László Lovász.

En 1998, la conjecture est à nouveau projetée sur le devant de la scène : un colloque est organisé à Paris sur ce thème[9] ; on apprend à peu près un an plus tard que le célèbre groupe formé autour de Paul Seymour de l'université de Princeton a décidé de considérer cette conjecture comme son nouveau terrain de chasse. Ces chercheurs sont connus pour leur excellence et leur persévérance dans la poursuite de longs projets.

La recherche a été encouragée par le support financier de l'American Institute for Mathematics, lui-même subventionné par la firme Fry Electronics. Si la somme allouée est comparable à celle offerte par la fondation Clay* pour la résolution d'un des sept problèmes ouverts, son mode d'attribution fut probablement plus constructif : Paul Seymour et ses collaborateurs ont été exonérés de tout enseignement, et leurs recherches étaient généreusement soutenues dans la première moitié de l'année 2001, et ce, indépendamment du résultat. Beaucoup de résultats importants utilisés dans la solution finale ont été démontrés par Vasek Chvatál et ses étudiants, ou encore par un autre groupe : Gérard Cornuéjols, Michele Conforti et leurs coauteurs.

Le 23 mai 2002, la naissance du théorème fort des graphes parfaits a été annoncée par Paul Seymour dans un colloque organisé au Tennessee par son collaborateur Robin Thomas (Georgia Tech, Atlanta). Les deux autres « chasseurs » qui ont participé à la résolution sont Neil Robertson (Ohio State University) et Maria Chudnovsky, une étudiante en thèse avec Seymour. Les auteurs estiment que la longueur de la démonstration va dépasser les 200 pages. Ils démontrent que les graphes de Berge se construisent en collant d'une manière « appropriée » des « graphes bipartis » (dans l'article : « à deux salles ») et d'autres graphes bien connus construits à l'aide de ces graphes. Leur méthode a de très bonnes chances d'aboutir aussi à un algorithme efficace (polynomial), mais cela demandera du travail supplémentaire. La date exacte de la naissance de cette conjecture est moins claire que la date de sa mort, ce qui est sûr c'est qu'elle a vécu une vie pleine et active pendant quarante-deux ans. Sa nouvelle vie comme théorème n'a peut-être pas fini de nous surprendre car elle pourrait conduire à d'autres démonstrations, applications ou algorithmes.

Pour en savoir plus

- Lewis Papadimitriou, « L'efficacité des algorithmes », dans *L'Intelligence de l'informatique*, Bibliothèque pour la science, 169-182, 1988.
- H. Wilf, *Algorithmes et complexité*, série « Logique mathématiques informatique », Masson, 1989.
- Cipra, *What's New in the Mathematical Sciences*, série de l'AMS.
- Le journal de la conjecture de Berge : www.cs.rutgers.edu/~chvatal/perfect/spgl.html
- La naissance du théorème fort des graphes parfaits : www.math.gatech.edu/~thomas/SLIDE/perfsl.pdf
- Et aussi : www.larecherche.fr