une multitude de formules extraordinaires qui n'ont parfois été démontrées qu'après sa mort.

Le programme HR s'est intéressé à l'arithmétique et a tenté à la fois de créer des concepts nouveaux – des définitions de suites numériques nouvelles –, de formuler des hypothèses à leur sujet et parfois de les démontrer à l'aide de programmes de démonstration automatique qu'il mettait à son service. Le programme disposait de critères pour ne se concentrer que sur des définitions potentiellement intéressantes. Il a pu ainsi identifier 14 suites numériques nouvelles, qui ont été acceptées en 2000 par l'Encyclopédie des suites numériques de Neil Sloane (http://oeis.org), où ne figurent que les suites ayant de l'intérêt aux yeux des mathématiciens.

L'une des suites proposées par HR est étonnamment simple et a conduit Simon Colton à publier un article dans le *Journal of Integer Sequences* en 1999. Il s'agit de la suite des «nombres refactorisables». Un entier n est refactorisable s'il est divisible par le nombre de ses diviseurs. L'entier 9 est par exemple refactorisable, car il a 3 diviseurs (1, 3 et 9) et que justement il est divisible par 3. Plus généralement, si p est premier, p^{p-1} est refactorisable.

À vrai dire, cette invention particulière du programme de Colton n'en n'était pas tout à fait une, car après la publication de l'article de 1999, on s'est aperçu qu'un autre article publié en 1990 avait envisagé la même notion… HR n'était donc pas le premier à s'intéresser à la notion, mais cela confirme que l'idée proposée est réellement jugée intéressante par les mathématiciens.

Notons que le programme de Simon Colton a découvert tout seul cette jolie propriété: «Si la somme des diviseurs d'un entier est un nombre premier, alors le nombre de diviseurs de ce nombre est aussi un nombre premier.»

Le programme HR a aussi inventé des notions concernant les groupes, les anneaux finis et les graphes. Il dispose de diverses mesures évaluant l'intérêt d'un concept, se concentre sur le plus prometteur et cherche alors à énoncer des propriétés le concernant pour lesquelles il tente de trouver des démonstrations ou des contre-exemples. Cette façon de procéder est proche de ce que font souvent les chercheurs en mathématiques, mais partiellement car un chercheur prend de la distance vis-à-vis de son sujet et tente d'établir des liens entre concepts appartenant à des champs éloignés.

UN VRAI SUCCÈS SUR LES GRAPHES

Le principe général de HR est particulièrement efficace quand il s'agit de structures finies, car le programme peut disposer d'une liste variée d'exemples et les utiliser pour tester si les propriétés qu'il envisage sont vérifiées par chaque élément de sa base d'exemples. Si c'est le cas, la propriété devient plausible et le système l'envisage comme conjecture. Si l'un des exemples >

4 LA MÉTHODE GRAFFITI

Le programme Graffiti énumère et sélectionne des conjectures portant sur les invariants des graphes.

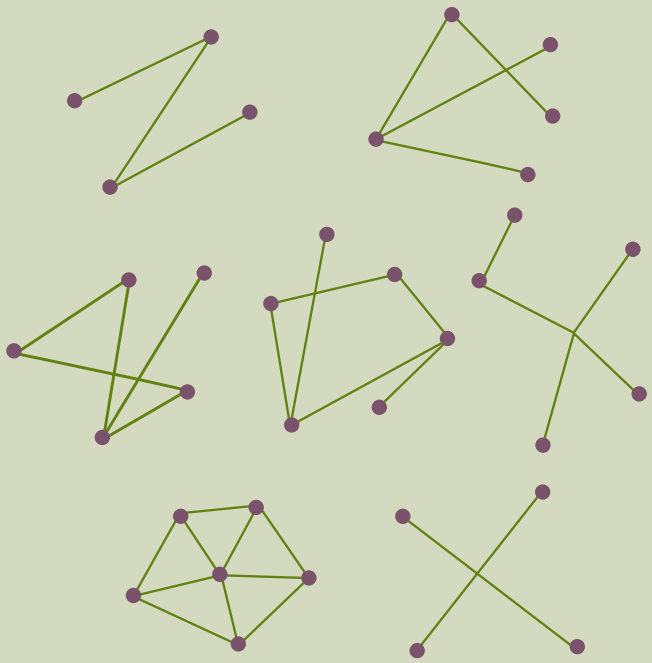
- Le programme considère les invariants inv_1, inv_2, …, inv_k et dispose pour chacun d'un sous-programme qui le calcule. Parmi les invariants possibles, il y a le nombre de nœuds du graphe, le nombre maximal de liens d'un nœud du graphe avec les autres nœuds, etc.

- Le programme dispose d'une base de test formée des graphes G_1, G_2, …, G_h. Cette base peut être complétée au fur et à mesure de l'utilisation du système pour le rendre plus efficace.

- Le programme dispose d'un système qui énumère les relations algébriques entre invariants. Ce système engendre par exemple $inv_1 + inv_2 \leq inv_3$, $inv_5/inv_4 > (inv_1)^2$, etc. Le système donnera la priorité aux relations simples.

- La recherche de conjectures portant sur les invariants consiste alors à tester les relations algébriques énumérées par le système. Celui-ci élimine toute relation non vérifiée par l'un des graphes de la base de test.

- Les relations algébriques passant les tests sont soumises à une seconde série de tests pour déterminer si elles sont connues ou résultent de relations connues. Les relations non écartées sont proposées comme conjectures aux mathématiciens, qui tentent de les démontrer.



Quelques exemples de graphes. Le système Graffiti s'appuie sur une base constituée de nombreux graphes pour tester des relations algébriques entre les «invariants» (nombre de nœuds d'un graphe, nombre moyen d'arêtes entre deux nœuds, etc.).

> contredit la propriété examinée, alors il est certain qu'elle est fausse, et il l'abandonne.

La théorie des graphes se prête bien à cette approche, et c'est pourquoi les méthodes automatiques de recherche de conjectures en théorie des graphes ont donné les premiers exemples incontestables de découvertes mathématiques faites par l'ordinateur.

Le programme Graffiti est la première grande réussite de cette approche. Il a été développé et utilisé de 1984 à 2002 par le mathématicien polonais Siemion Fajtlowicz.

Graffiti s'intéresse particulièrement aux invariants des graphes. Ce sont par exemple le diamètre du graphe (la distance – en nombre d'arêtes – maximale séparant deux nœuds), le nombre chromatique d'un graphe (le nombre minimal de couleurs qu'il faut pour colorer les nœuds du graphe en évitant que deux nœuds voisins soient de même couleur), le degré maximal d'un graphe (nombre maximal de voisins d'un nœud), etc.

Plusieurs dizaines d'invariants sont étudiés et les relations entres ces invariants sont complexes, mais essentielles pour la théorie.

En menant des recherches automatisées de relations entre ces invariants, Graffiti a proposé de nombreuses conjectures intéressantes. Le programme élimine toutes les conjectures contredites par un des graphes qu'il connaît, et celles classiques ou s'en déduisant facilement. Siemion Fajtlowicz indique d'ailleurs que c'est ce travail de sélection qui a été le plus délicat et complexe à programmer. Parmi les conjectures de Graffiti, certaines ont été démontrées par la suite ou invalidées par des chercheurs de la discipline, donnant lieu à plusieurs publications scientifiques et thèses universitaires.

Il ne fait pas de doute que ce programme et ses successeurs contribuent réellement aux progrès mathématiques. Siemion Fajtlowicz a d'ailleurs été amené à travailler avec les meilleurs spécialistes de la théorie des graphes comme Paul Erdős, Bela Bollobas ou Paul Spencer. Erdős et ses coauteurs, dans un article de 1988 où ils démontrent l'une des conjectures du programme Graffiti, reconnaissent son apport. La conjecture en question affirme que $dm_G \leq r_G$, où dm_G est la distance moyenne entre deux nœuds du graphe G, et r_G est la somme des inverses du nombre de voisins des nœuds de G.

En reprenant le principe de Graffiti, d'autres thèmes de recherche ont été abordés et ont donné lieu à une interaction entre mathématiciens humains et ordinateurs proposant conjectures, réfutations et parfois démonstrations. Aujourd'hui encore, en théorie des graphes et dans certains autres domaines, l'utilisation de générateurs de conjectures aide les chercheurs.

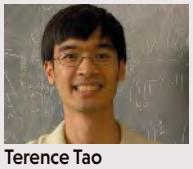
RECHERCHER DES FORMULES

Un événement remarquable s'est produit en 1995 dans le domaine de l'aide informatique à la recherche mathématique. Simon Plouffe a programmé sa machine pour qu'elle explore des formules d'un certain type donnant le nombre π et en a effectivement trouvé une que personne ne soupçonnait. Ici, les objets manipulés ne sont pas finis, puisqu'il s'agit de nombres réels ayant une infinité de décimales. Même si la méthode utilisée pour les conjectures sur les graphes ne s'applique pas, le procédé utilisé lui ressemble.

On se donne plusieurs nombres réels dont on connaît des évaluations aussi précises qu'on le veut ; c'est le cas pour π, et pour les sommes de séries dont le terme général est de la forme $1/[r^k(ak+b)]$. On recherche alors s'il existe une combinaison linéaire à coefficients entiers entre les nombres choisis qui soit nulle avec quelques décimales de précision. C'est un travail patient de calcul impossible à mener à la main, mais pour lequel on dispose d'algorithmes efficaces. Quand une relation est trouvée, on teste si elle reste vraie avec un plus grand nombre de décimales. Là encore, c'est

5 L'ORDINATEUR MATHÉMATICIEN ?

Voici les réponses données en 2014 par de célèbres mathématiciens à la question : « Dans cent ans ou dans mille ans, les ordinateurs dépasseront-ils les humains en mathématiques comme ils les dépassent maintenant au jeu d'échecs ? »
Terence Tao (médaille Fields en 2006) : « Les ordinateurs vont certainement devenir encore plus puissants, mais j'espère que l'essentiel du travail mathématique continuera d'être fait par des humains… utilisant des ordinateurs. »
Richard Taylor (ce professeur à l'université Harvard a aidé Andrew Wiles à démontrer le grand théorème de Fermat) : « Une meilleure question serait : est-ce qu'un ordinateur recevra la médaille Fields ? »
Jacob Lurie (professeur à l'IAS de Princeton) : « Mille ans, c'est une durée trop longue pour faire des prévisions. »
David Ruelle (professeur à l'Institut des hautes études scientifiques) envisage une sorte de passage de témoins de l'homme à la machine pour faire des mathématiques : « La capacité intellectuelle à faire des mathématiques est un développement récent du point de vue de l'évolution biologique. J'ai du mal à croire que ce développement récent ait produit quelque chose de si unique qu'il ne peut être imité avec succès par les ordinateurs. Je pense plutôt que la créativité mathématique humaine […] est une étape dans l'évolution, et qu'il y aura d'autres étapes. La grande question devient alors : quel genre de mathématiques pourrait être produit par une créativité mathématique artificielle ? »
Marcus du Sautoy (professeur à Oxford), dans son livre *The Creativity Code* (2019), indique : « Les méthodes d'apprentissage automatique donnent aujourd'hui des résultats décevants en mathématiques. Seront-elles un jour capables de s'appliquer aux mathématiques que nous aimons et d'en créer qui leur ressemblent ? C'est bien possible et l'échec actuel est probablement simplement un sursis. »


Terence Tao


Richard Taylor


Jacob Lurie


David Ruelle


Marcus du Sautoy

une question de calcul. Si la relation est numériquement confirmée, l'algorithme propose la conjecture. En utilisant alors les méthodes classiques, le mathématicien reprend la main et tente de démontrer l'égalité.

C'est ainsi que la formule suivante fut découverte, puis rapidement démontrée:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right)$$

L'intérêt de cette formule est qu'elle permet de calculer les chiffres binaires de π à un endroit donné du développement binaire, sans avoir à calculer les chiffres qui précèdent. Incontestablement, l'ordinateur avait découvert un résultat intéressant. Grâce à ce type de formules, on connaît aujourd'hui des chiffres binaires de π autour de la position 4×10^{16} alors que les méthodes qui calculent tout ne permettent pour l'instant que d'atteindre la position 3×10^{14}.

TROUVER DES DÉMONSTRATIONS

Un certain nombre de tentatives ont été faites pour introduire dans le domaine des mathématiques des méthodes de nature statistique et neuronale.

Kshitij Bansal et son équipe, au centre de recherche de Google à Mountain View, en Californie, a mis au point un programme, intitulé HOlist, pour démontrer des théorèmes en se fondant sur une méthode d'apprentissage. Partant du système d'aide à la formalisation et au contrôle des démonstrations HOL-Light, qui a été utilisé pour la mise au point de la démonstration de la conjecture de Kepler en 2014, les chercheurs ont programmé un outil automatique qui recherche des démonstrations en s'inspirant d'une base de démonstrations fournies en exemples. Le système a eu accès à plus de 10000 démonstrations de théorèmes, exprimées dans le langage de HOL-Light. Parmi elles de nombreuses démonstrations de résultats intermédiaires de la preuve de la conjecture de Kepler.

Notons que ces démonstrations avaient presque toujours été mises au point par un travail guidé par des mathématiciens. Grâce aux paramètres du réseau de neurones du système HOlist, ajustés par des milliers d'exemples, celui-ci sait mieux comment choisir et ordonner les éléments d'une démonstration pour atteindre un but. On lui a resoumis les énoncés des théorèmes ayant servi à son apprentissage, et il a pu retrouver, grâce aux tactiques de démonstration apprises, et cette fois seul sans guidage par des mathématiciens, la démonstration de 58% d'entre eux. Plus intéressant, quand on lui a soumis 3217 énoncés pris en dehors des énoncés de la base d'exemples, et qu'on a demandé au système de les démontrer, il a su le faire pour 1251 d'entre eux. Mais pour l'instant, le système n'a pas découvert de théorèmes nouveaux.

CALCULER DES PRIMITIVES

Un autre exemple tout récent d'utilisation de l'apprentissage profond concerne le calcul des primitives en analyse et la résolution d'équations différentielles. Certaines méthodes utilisées par les mathématiciens, par exemple l'intégration par parties, ou le changement de variable, sont programmées pour qu'un système réussisse seul des calculs d'intégrales. Beaucoup de travaux ont été ainsi menés conduisant à des programmes de plus en plus complexes et efficaces. En reprenant le problème avec des méthodes d'apprentissage et des réseaux de neurones, Guillaume Lample et François Charton, de Facebook, ont mis au point une méthode différente, fondée sur la présentation d'exemples à un réseau de neurones soigneusement configuré pour apprendre sur ce type de problèmes symboliques.

Les exemples proposés au système étaient de trois types: (1) des paires $[f, F]$, où F est une primitive de f, provenant des programmes de calcul fondés sur l'algorithme de Risch ou une de ses variantes; (2) des paires provenant de dérivation qui, elle, est facile et maîtrisée depuis longtemps, contrairement au calcul de primitives: on dérive F, ce qui donne f, d'où la paire $[f, F]$; (3) des paires exploitant la méthode d'intégration par parties.

Les performances du système obtenu égalent et parfois dépassent celle des systèmes classiques comme Mathematica et Matlab. Cela montre de nouveau que l'association de méthodes purement symboliques (dans la phase d'apprentissage) et de méthodes neuronales fait progresser la solution de problèmes symboliques.

L'HUMAIN RESTE INÉGALÉ

Ainsi, pour certaines tâches spécialisées, les programmes égalent ou battent les humains, mais ces méthodes ont toujours des champs de réussite étroits. Même si, indubitablement, la recherche mathématique en tire une aide, les systèmes conçus aujourd'hui se bloquent et cessent de progresser.

Il est formidable d'avoir développé des systèmes informatiques qui formulent d'hypothétiques relations entre les paramètres des graphes, qui mènent des calculs arithmétiques et en tirent des formules de séries nouvelles, qui effectuent des calculs de primitives ou proposent quelques concepts inattendus par combinaison.

Cependant, tous atteignent vite leurs limites. Au contraire, les mathématiciens bouillonnent d'idées nouvelles que les ordinateurs aident parfois à maîtriser, mais en restant toujours de simples esclaves calculateurs sans véritable créativité. L'ordinateur authentiquement mathématicien n'est pas encore là, on attend sa venue! ■

BIBLIOGRAPHIE

An environment for machine learning of higher-order theorem proving, *Proc. of the 36th International Conference on Machine Learning*, 2019.

G. Lample et F. Charton. **Deep learning for symbolic mathematics**, prépublication arXiv:1912.01412, 2019.

C. E. Larson et N. Van Cleemput, **Automated conjecturing I : Fajtlowicz's Dalmatian heuristic revisited**, *Artificial Intelligence*, vol. 231, pp. 17-38, 2016.

D. B. Lenat et P. J. Durlach, **Reinforcing math knowledge by immersing students in a simulated learning-by-teaching experience**, *International Journal of Artificial Intelligence in Education*, vol. 24, pp. 216-250, 2014.

C. E. Larson, **A survey of research in automated mathematical conjecture-making**, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 69, pp. 297-316, 2005.

S. Colton, **Automated Theory Formation in Pure Mathematics**, Springer, 2002.

S. Colton, **Refactorable numbers – a machine invention**, *Journal of Integer Sequences*, vol. 2., article 99.1.2, 1999.