par Nicolas K. Blanchard

Prouver rapidement qu'une propriété est vérifiée… ou pas

Peut-on donc créer des algorithmes probabilistes qui décident rapidement si un objet mathématique satisfait une propriété donnée ou s'il en est loin ? Le but des testeurs de propriété est de décider entre ces deux situations.

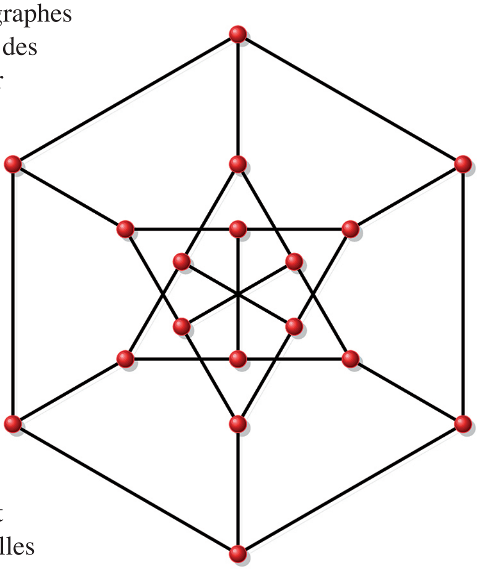
La cathédrale de Meaux.

L'image ci-contre provient-elle d'une photo d'une cathédrale ? Notre cerveau nous dit que oui, *a priori*, et ce sans faire vraiment attention aux détails de l'image. Maintenant considérons le graphe ci-contre, est-il possible de le colorier avec trois couleurs (de manière que deux sommets quelconques reliés par une arête soient de couleurs différentes) ? Il nous faut *a priori* regarder en détail tout le graphe, et même faire plusieurs essais, pour le savoir. Un *testeur de propriété* est un algorithme probabiliste qui permet de différencier avec bonne probabilité entre le cas où l'entrée a une certaine propriété P, et le cas où l'entrée est loin d'avoir cette propriété P. Dans le cas de la cathédrale, le testeur devrait répondre oui si on a une cathédrale, et non si on a une image de tortue, mais peut se tromper quand l'image représente une construction architecturale. Dans le cas du graphe, le testeur doit répondre non quand il faut enlever au moins une proportion constante des arêtes pour rendre le graphe coloriable.

Dépendre le moins possible de l'entrée

Peut-on donc créer des algorithmes probabilistes qui décident rapidement si l'entrée satisfait une propriété P ou si elle en est loin ? Pour fixer les idées, peut-on disposer d'un algorithme répondant oui avec probabilité 2 / 3 quand P est vraie, et non avec probabilité 2 / 3 quand il faut changer une proportion ε de l'entrée pour qu'elle satisfasse P (sans se soucier de ce qui passe quand l'entrée ne satisfait pas P mais n'en est pas loin) ? Pour les problèmes usuels, il est souvent aisé de répondre (et même sans commettre d'erreur), en ayant accès à toute l'entrée, mais le but des testeurs de propriétés est de dépendre le moins possible, voire pas du tout, de la taille de l'entrée ! Ainsi, vérifier qu'un graphe est 3-coloriable peut être fait sans dépendance en la taille du graphe mais seulement en ε si l'on considère qu'un graphe est loin d'être 3-coloriable quand on doit lui enlever $(\varepsilon / 2)\, n^2$ arêtes pour le rendre 3-coloriable, soit un facteur ε des arêtes possibles.

Considérons une fonction $f : \{0, 1\}^n \to \{0, 1\}$, donnée par le tableau complet de toutes ses valeurs. On se demande si f est linéaire, c'est-à-dire si $f(x + y) = f(x) + f(y)$ pour tous x et y. Vérifier cela pour chaque paire (x, y) prendrait $(2^n)^2$ opérations : ce n'est pas raisonnable. C'est cependant faisable si l'on suppose qu'elle est ou bien linéaire, ou bien loin de l'être, c'est-à-dire qu'il faudrait changer au moins $(\varepsilon / 2) \times 2^n$ valeurs de f pour la rendre linéaire. Dans ce cas, un algorithme naïf fonctionne très bien : on choisit une paire (x, y) uniformément au hasard, et on vérifie que $f(x + y) = f(x) + f(y)$. On répète ce test $1 / \varepsilon$ fois, et on décide que la fonction est linéaire si, et seulement si, elle passe tous les tests.
Si f est linéaire, elle passera tous les tests, donc la probabilité d'erreur sera 0. Si f est loin d'être linéaire, il est possible, mais non trivial, de montrer que la probabilité que le test prouve la non-linéarité est au moins $\varepsilon / 6$, prouver la non-linéarité correspondant à trouver a et b tels que $f(a + b) \neq f(a) + f(b)$. Ainsi, la probabilité de se tromper sur tous les tests quand f est loin d'être linéaire est au plus $(1 - \varepsilon / 6)^{1/\varepsilon} < e^{-1/6}$. En augmentant le nombre de tests, on peut rendre cette probabilité d'erreur arbitrairement petite. C'est là un exemple sympathique, car la dépendance en ε est en pratique souvent bien pire, à cause d'un célèbre résultats, le *lemme de régularité de Szemerédi*, selon lequel tout graphe de taille suffisante peut être partitionné en un nombre borné de sous-graphes tels que les arêtes entre sous-graphes ressemblent (à ε près) à des arêtes aléatoires. Pour de nombreuses propriétés, tester si le graphe de départ les possède est équivalent à les tester sur le graphe des sous-graphes, qui est de taille bornée par une fonction de $1 / \varepsilon$. Le problème est que cette fonction est souvent une tour d'exponentielles en $1 / \varepsilon$.



Le graphe de Pappus.

Un intérêt renforcé aujourd'hui

Pour le test de linéarité, un atout supplémentaire apparaît : si la fonction est linéaire ou proche d'être linéaire, le testeur peut la retrouver en utilisant exactement la même technique. Cela a des applications dans les codes correcteurs d'erreurs, et dans la preuve théorème PCP (voir dans ce dossier). Ces algorithmes peuvent donc être très puissants, même si la dépendance en ε peut sembler prohibitive. S'ils concernaient surtout des problèmes de graphe il y a vingt ans, on les trouve maintenant dans des applications d'apprentissage et de *big data*. Un programme basé sur cette théorie rivalisait récemment avec les meilleurs logiciels de reconnaissance d'images, et l'explosion de la taille des données à traiter par rapport aux capacités actuelles de calcul ne peut que renforcer l'intérêt qui leur est porté.

N.K.B.