L'art d'écrire des programmes

Christine Solnon

L'art d'écrire des programmes

Christine Solnon, CITI, Inria, INSA Lyon, F-69621 Villeurbanne

Chronique parue dans La recherche N° 572, janvier-mars 2023

On n'a jamais fini d'apprendre à programmer. Telle pourrait être la devise de l'informaticien américain Donald E. Knuth, qui vient de publier le volume 4B de *The Art Of Computer Programming*, également connu sous l'acronyme TAOCP. Né en 1938, Knuth n'en est pas à son coup d'essai ; TAOCP est en effet une véritable saga de l'informatique qui devait comporter sept volumes selon un plan conçu en 1962. Les trois premiers, publiés en 1968, 1969 et 1973, sont des ouvrages fondateurs de la science des algorithmes et ont valu à l'auteur de recevoir le prestigieux prix Turing en 1974. Le volume 4 devait être consacré aux algorithmes combinatoires, mais le sujet s'est révélé trop riche et il a été découpé en six volumes numérotés de 4A à 4F. Le premier (volume 4A), paru en 2011, s'intéresse à l'énumération de combinaisons. Le deuxième (volume 4B), qui vient de paraître, étudie l'exploration d'espaces de recherche.

Pourquoi explorer un espace de recherche ? Car cela permet de résoudre de nombreux problèmes très sérieux : concevoir un emploi du temps satisfaisant des contraintes, planifier des trajectoires en évitant les collisions, ou ranger des caisses dans des conteneurs en minimisant l'espace perdu. Dans son livre, Donald E. Knuth met en scène les algorithmes de façon ludique, en utilisant des énigmes et puzzles mathématiques tels que la création de grilles de mots croisés, la résolution de Sudoku ou le placement de polyominos [1]. Pour résoudre de tels problèmes combinatoires, il s'agit d'explorer méthodiquement un espace de recherche contenant des combinaisons candidates jusqu'à trouver une combinaison valide, ou alors prouver qu'il n'existe pas de combinaison valide.

Énumérer un ensemble de combinaisons est un problème facile à résoudre en utilisant la récursivité. Par exemple, pour générer toutes les grilles possibles d'un Sudoku comportant n cellules, on liste toutes les valeurs possibles

1. Le polyomino est une forme géométrique constituée de carrés identiques qui partagent une arête. Le jeu de Tétris, par exemple, utilise des polyominos de quatre carrés appelés tétrominos.

pour la première cellule puis, pour chacune de ces valeurs, on génère toutes les combinaisons possibles pour les $n-1$ cellules restantes. Alors pourquoi consacrer plusieurs volumes de TAOCP à ce sujet ? La difficulté vient du fait qu'il existe généralement un nombre exponentiel de combinaisons, de sorte qu'on est confronté à un phénomène d'explosion combinatoire : les petits problèmes sont vite résolus, mais une légère augmentation de la taille du problème rend impossible l'énumération exhaustive en un temps raisonnable. Aussi doit-on introduire des raisonnements permettant d'éliminer a priori des ensembles de combinaisons. Ainsi, pour le Sudoku, on peut maintenir pour chaque cellule une liste de valeurs candidates et filtrer ces listes en propageant les contraintes de différence : lorsqu'une liste devient vide, on revient en arrière pour essayer une autre valeur. Ces techniques ont souvent des effets impressionnants en pratique, comme cela est souligné par Donald E. Knuth dans sa préface : *L'art d'écrire de tels programmes est particulièrement important et attrayant car une seule bonne idée peut faire gagner des années, voire des siècles, de temps de calcul.*

Un point critique réside dans le fait que les filtrages doivent être annulés lors des retours en arrière (de la même façon que, lorsqu'on résout un Sudoku, on note au crayon les valeurs incompatibles avec les hypothèses en cours et on gomme ces valeurs quand une hypothèse se révèle mauvaise). Le volume 4B introduit les *dancing links*, une structure de données qui permet de supprimer les cellules d'une matrice puis de les restaurer lors des retours en arrière dans une danse épatante, à la fois facile à mettre en œuvre et très efficace. Les *dancing links* sont utilisés pour résoudre le problème de la couverture exacte, l'un des vingt et un problèmes NP-complets répertoriés par l'informaticien américain Richard Karp en 1972. Ces problèmes sont assez déroutants car, s'il est facile de vérifier qu'une solution donnée est correcte, il est bien plus difficile de trouver une solution ou de prouver qu'il n'en existe aucune. Knuth remet le problème de la couverture exacte sur le devant de la scène, en montrant qu'il s'agit d'une généralisation de très nombreux autres problèmes qui peuvent être résolus avec des *dancing links*.

La dernière partie du volume 4B est consacrée au problème SAT. Ce problème étonnamment simple dans sa formulation (il s'agit de décider si une formule de logique propositionnelle admet une solution) est très célèbre. En effet, il s'agit du premier problème qui a été démontré NP-complet, par l'américano-canadien Stephen Cook en 1971, et les avancées spectaculaires réalisées ces dernières années ont permis de concevoir des solveurs SAT, utilisés pour résoudre de multiples problèmes tels que la planification ou la conception de circuits intégrés. Knuth décrit et analyse ces différentes techniques, qu'elles soient anciennes ou très récentes, en les replaçant dans leur

contexte historique. Il ouvre également un nombre phénoménal de nouvelles pistes, notamment à travers les exercices, souvent ludiques, parfois inattendus, toujours instructifs, et dont les solutions sont de véritables trésors.

Les raisonnements utilisés pour contenir l'explosion combinatoire lors de la résolution de problèmes NP-complets sont des techniques d'intelligence artificielle étudiées depuis des décennies. Mais pourrait-on utiliser des techniques plus récentes, comme l'apprentissage profond (le *deep learning*), pour résoudre des problèmes NP-complets ? Cette question est légitime dans la mesure où des tâches complexes sont maintenant résolues par apprentissage (jouer au go ou identifier des personnes dans des vidéos, par exemple). Étant donné l'importance des applications des problèmes NP-complets, cette piste de recherche est activement explorée, et des travaux récents montrent qu'on peut apprendre des heuristiques (c'est-à-dire, des règles empiriques dont l'efficacité n'est pas garantie). Cependant, une première limite de ces travaux est que l'apprentissage se fait en exploitant les solutions d'un grand nombre de problèmes similaires. Si ces problèmes ne sont pas suffisamment nombreux, ou trop différents des problèmes à résoudre, les heuristiques apprises ne sont pas pertinentes. Une seconde limite est que, si les heuristiques apprises peuvent parfois aider à trouver plus rapidement une solution, elles ne permettent pas de prouver l'absence de solution.

Enfin, signalons que Knuth, un brin facétieux, récompense par un chèque de 2,56 dollars toute erreur trouvée dans ses livres. Ces chèques sont émis par la *Bank of San Serriffe*, une banque fictive dont les soldes des comptes sont publics, ce qui crée une petite émulation pour apparaître parmi les plus grandes fortunes de cette banque !