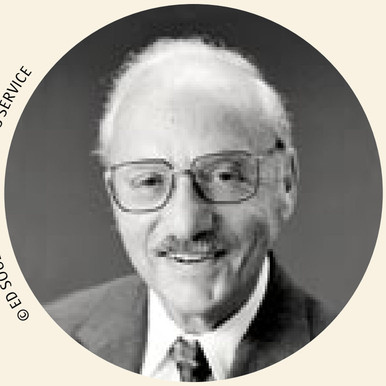
W^xyz

# Programmation linéaire





Quand un ingénieur se trouve face à un problème trop complexe pour être traité même avec le plus puissant des ordinateurs, il doit biaiser. L'une des solutions est de faire appel à un algorithme reposant sur le découpage de polyèdres.

# Des solutions pour faire bonne figure

Depuis les travaux du mathématicien canadien Stephen Cook, au début des années 1970, on sait qu'il existe des problèmes d'optimisation que l'on ne peut pas toujours résoudre en un temps « raisonnable ». Dans la pratique, cela signifie que, certes on sait comment faire pour trouver la solution, certes on peut s'aider des ordinateurs pour effectuer les calculs, mais, aussi puissants ces ordinateurs soient-ils, il leur faudrait des milliards d'années pour donner une solution (lire le dossier, p. 30). Il s'agit des problèmes « NP-difficiles ». Le problème du voyageur de commerce en est l'archétype : étant donné un ensemble de villes et, pour chaque couple de villes, la distance qui les sépare, trouver une tournée (un circuit passant une et une seule fois par chacune des villes) de longueur minimale.

Ces problèmes ne sont pas que des casse-tête de mathématiciens. On les rencontre en informatique (répartition de tâches entre des serveurs), dans les télécommunications (affectation de fréquences dans les réseaux GSM), le transport (emploi du temps des personnels navigants d'une compagnie aérienne), la finance (constitution d'un portefeuille d'actions), etc. Comment procéder en pratique, quand l'ingénieur a absolument besoin d'une solution ? Dans les années 1950, trois mathématiciens américains, George Dantzig, Ray Fulkerson et Selmer Johnson, ont commencé à développer des méthodes originales, dites « polyédriques ». Elles ont été ensuite reprises et développées par le mathématicien canadien Jack Edmonds.

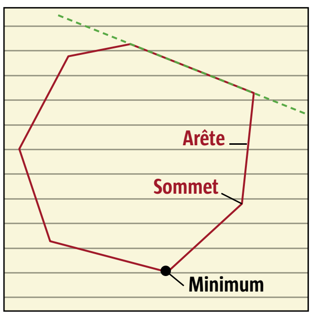
**Renaud Sirdey,**
**ingénieur de recherche chez Nortel et doctorant dans l'unité heuristique et diagnostic des systèmes complexes (CNRS-université de technologie de Compiègne).**
renauds@nortel.com

## Le « simplexe »

Elles reposent sur la notion de « programmation linéaire ». Ce problème consiste à optimiser, c'est-à-dire trouver le minimum ou le maximum d'une fonction linéaire (de la forme $c_1x_1 + ... + c_nx_n$, où les $c_i$ sont les coefficients, donnés, et les $x_i$ les variables à déterminer) en respectant un ensemble de contraintes linéaires (de la forme $a_1x_1 + ... + a_nx_n \geq b$).

En 1947, George Dantzig (en médaillon) a réalisé ce que l'on considère aujourd'hui comme l'un des tours de force des mathématiques appliquées du XX^e siècle en inventant l'algorithme du « simplexe », qui permet de résoudre le problème de la programmation linéaire. Cet algorithme a immédiatement trouvé un usage pratique : on raconte que, dès les années 1950, une compagnie aérienne américaine a instantanément amorti l'achat de l'un des tout premiers ordinateurs d'IBM en réorganisant l'emploi du temps de ses équipages selon la solution d'un programme linéaire.

**Fig.1 Optimiser sur un polyèdre**




**UN POLYÈDRE peut être vu comme un ensemble de segments, fragments de droites coupant le plan en deux : c'est donc l'intersection d'un nombre fini de demi-espaces fermés. Optimiser revient à trouver le point le plus bas sur des « lignes de niveaux ». Ce point est un sommet du polyèdre.**



## Équation nutritionnelle

Un autre exemple, traité à la fin des années 1940 : comment déterminer le régime alimentaire le moins coûteux qui soit conforme aux recommandations des nutritionnistes ? Voici le raisonnement. On dispose d'un ensemble de denrées alimentaires et, pour chacune d'entre elles, du coût unitaire, ainsi que d'un ensemble de valeurs nutritionnelles (apports unitaires en glucides, protides, lipides, etc.). Il s'agit de minimiser le coût total (les $x_i$ sont les quantités achetées et les $c_i$ les coûts unitaires), tout en respectant les apports recommandés pour chacun des nutriments (les $a_i$ sont les apports des denrées pour le nutriment). La résolution de cette question selon un calcul réalisé à la main au bureau américain des normes a permis de voir qu'à cette époque il était possible de se nourrir correctement pendant un an pour un peu moins de 40 dollars.

## Le meilleur sommet

Si l'on raisonne maintenant en termes de géométrie, le problème de la programmation linéaire revient à optimiser une fonction linéaire sur un polyèdre. On montre que, s'il existe, l'optimum est réalisé en l'un des sommets du polyèdre [fig. 1].

L'algorithme du simplexe consiste à partir d'un sommet du polyèdre puis, à chaque itération, à emprunter l'une de ses arêtes de manière à atteindre un sommet voisin meilleur. On répète l'opération jusqu'à ce que le sommet atteint soit au moins aussi bon que tous ses voisins, donc optimal.

Ajoutons maintenant une contrainte supplémentaire au problème de la programmation linéaire : certaines variables doivent prendre des valeurs entières, c'est la « contrainte d'intégrité » — il s'agit alors de programmation linéaire en nombres entiers. Par exemple, dans le cas du problème du voyageur de commerce, on introduit les variables $x_{ij}$ telles que $x_{ij} = 1$ si et seulement si le voyageur emprunte l'arête reliant la ville *i* à la ville *j*, et $x_{ij} = 0$ sinon. Avoir affaire exclusivement à des entiers nous évite de voir le voyageur ne se déplacer qu'à moitié d'une ville à une autre… En pratique, on constate que ce formalisme permet d'exprimer presque tous les problèmes NP-difficiles.

Si l'on revient à la géométrie, la contrainte d'intégrité ne change pas la nature du problème : il s'agit toujours d'optimiser une fonction linéaire sur un polyèdre — ce dernier s'appelant alors polyèdre entier. Malheureusement, pour les problèmes NP-difficiles, la structure de ce polyèdre est d'une complexité à toute épreuve. Et, pour le décrire, il faut avoir recours à un nombre de contraintes astronomique – il en va de même du nombre de chiffres requis pour représenter les coefficients, les $a_i$, de bon nombre de ces contraintes.

On se contente alors de retenir des informations très partielles (un petit nombre de contraintes pertinentes) sur la structure du polyèdre entier. Une approche parmi les plus performantes pour la résolution exacte des problèmes NP-difficiles consiste à partir du programme linéaire obtenu en ignorant la contrainte d'intégrité, puis à l'enrichir itérativement de facettes (faces de dimension maximale) du polyèdre entier, de manière à « couper » les solutions fractionnaires successives [fig. 2] – on parle d'algorithme de coupes. Le travail est terminé lorsque l'on obtient une solution entière lors d'une itération : c'est la solution optimale.

En résumé, on cherche autant que faire se peut, à substituer la résolution du problème en nombres entiers, qui est difficile, par celle d'une séquence de programmes linéaires de petite taille, qui est facile. Dans la mesure où notre connaissance du polyèdre entier n'est que partielle, cette méthode ne marche pas à tous les coups. Il arrive que l'algorithme s'arrête avec une solution fractionnaire, inexploitable. Dans ce cas, il faut se résoudre à explorer l'ensemble des solutions en décomposant le problème en sous-problèmes de manière (généralement) récursive.

**TROUVER LE PLUS COURT CHEMIN pour relier des villes en passant une seule fois par chacune d'entre elles : tel est le fameux problème dit du voyageur de commerce, qui est de mieux en mieux résolu lorsque le nombre de villes augmente.** © GEOEYE/SPL/COSMOS

## Algorithme de coupes

Pour tester de nouvelles approches algorithmiques, rien de tel que – encore ! – le problème du voyageur de commerce. Déjà, dans les années 1950, Dantzig, Fulkerson et Johnson avaient réussi à résoudre une instance à 49 villes à l'aide d'un algorithme de coupes. Plus récemment, c'est à l'aide d'un algorithme de ce type, mais particulièrement raffiné, et d'une bonne dose d'ingéniosité informatique, qu'une équipe nord-américaine a réussi à résoudre une instance du problème à 24 978 villes !

Des algorithmes de ce type permettent aux ingénieurs de s'attaquer à des problèmes industriels concrets. Mais ils ne peuvent pas toujours obtenir de solution exacte lorsque la taille des instances augmente. Il est alors généralement possible de prouver que la valeur de la meilleure solution rencontrée est très proche de celle d'une solution optimale. Une solution approchée, certes, mais une bonne approximation ! ■

**Fig.2 Relaxation linéaire**

**LORSQUE LA STRUCTURE du polyèdre entier est trop complexe, on ignore la contrainte d'intégrité : le polyèdre P correspond à la relaxation linéaire du polyèdre entier $P_I$. L'algorithme de coupe donne de façon itérative des droites F, qui permettent d'écarter les sommets fractionnaires x. On retrouve ainsi des facettes de P.**