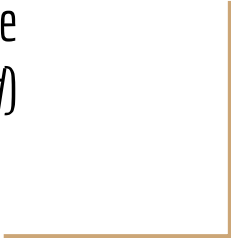




Voyageur de commerce

Algorithme de Little
(*Branch and bound*)

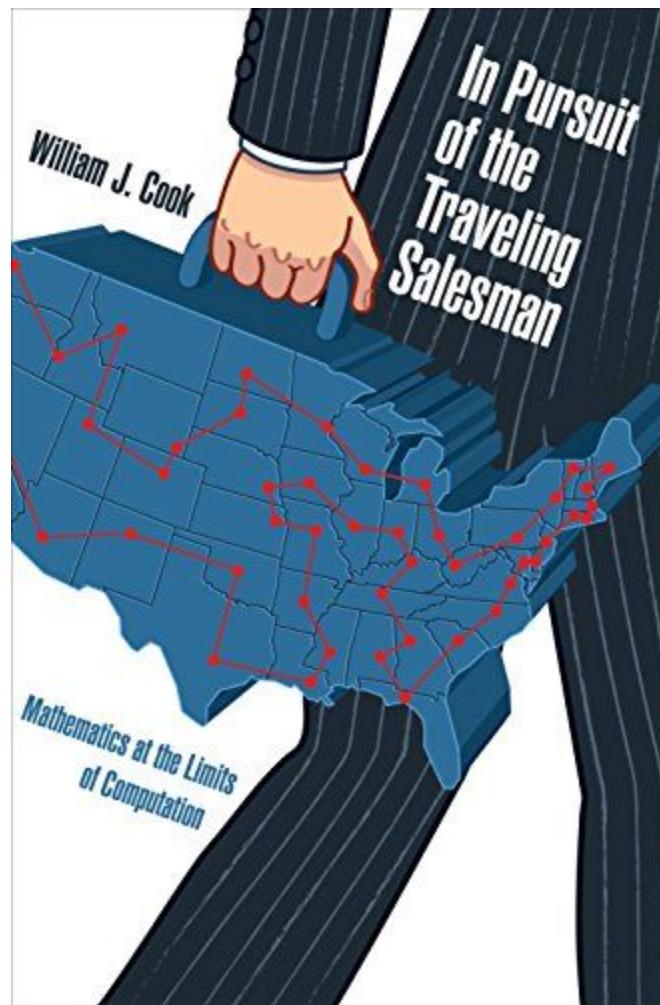


Présentation

Le problème du voyageur de commerce est un incontournable de la recherche opérationnelle. Il appartient à la classe des problèmes complexes en optimisation combinatoire et sa proximité avec les problèmes d'affectation en fait un problème passionnant pour saisir la difficulté intrinsèque de la combinatoire. Nous allons réutiliser la partie réduction de la méthode hongroise vue pour l'affectation afin de résoudre le problème du voyageur de commerce via l'algorithme de Little (1963), basé sur les principes de séparation et évaluation (branch and bound).

Modélisation

Le problème du voyageur de commerce (*Traveling Salesman Problem* : TSP) est un problème d'optimisation qui consiste à trouver un circuit hamiltonien de coût minimal dans un graphe orienté valué représentant des distances entre villes.



Complexité

Le TSP appartient aux problèmes NP-complets, pour laquelle aucun algorithme efficace n'a été trouvé. Les seuls algorithmes exacts (garantissant la minimalité de la solution) connus jusqu'à présent sont donc tous de complexité exponentielle.

Deux approches s'offrent alors à nous, soit flirter avec les limites afin de garantir la minimalité, soit rechercher des solutions approchées via des algorithmes efficaces.

Méthodes exactes

La première méthode exacte à considérer est bien entendu la recherche exhaustive consistant à rechercher tous les circuits hamiltoniens et à garder celui de coût minimum.

Cette méthode atteint très rapidement des limites non exploitables car de complexité en $O(n!)$. A titre d'exemple pour 20 villes, cela nécessite

$$20! = 2\,432\,902\,008\,176\,640\,000$$

soit plus de 2 milliards de milliards d'instructions machines. Même avec les meilleurs calculateurs actuels cette limite reste inatteignable.

Méthodes exactes (suite)

Deux améliorations permettent de réduire drastiquement cette limite bien que restant exponentiels.

La première utilise un algorithme de programmation dynamique que l'on doit à [M. Held et M. Karp en 1962](#), qui résout TSP en $O(n^2 * 2^n)$, ce qui ramène notre problème avec 20 villes dans des limites acceptables : 419 430 400 instructions machines.

Méthodes exactes (fin)

La seconde utilise un algorithme de séparation et évaluation que l'on doit à [J.D.C Little, K.G. Murty, D.W. Sweeney et C. Karel en 1963.](#)

Bien que sa complexité soit difficile à déterminer, il s'agit à l'heure actuelle de la solution exacte la plus efficace expérimentalement, certains problèmes de plusieurs centaines de villes ayant été résolus en l'utilisant.

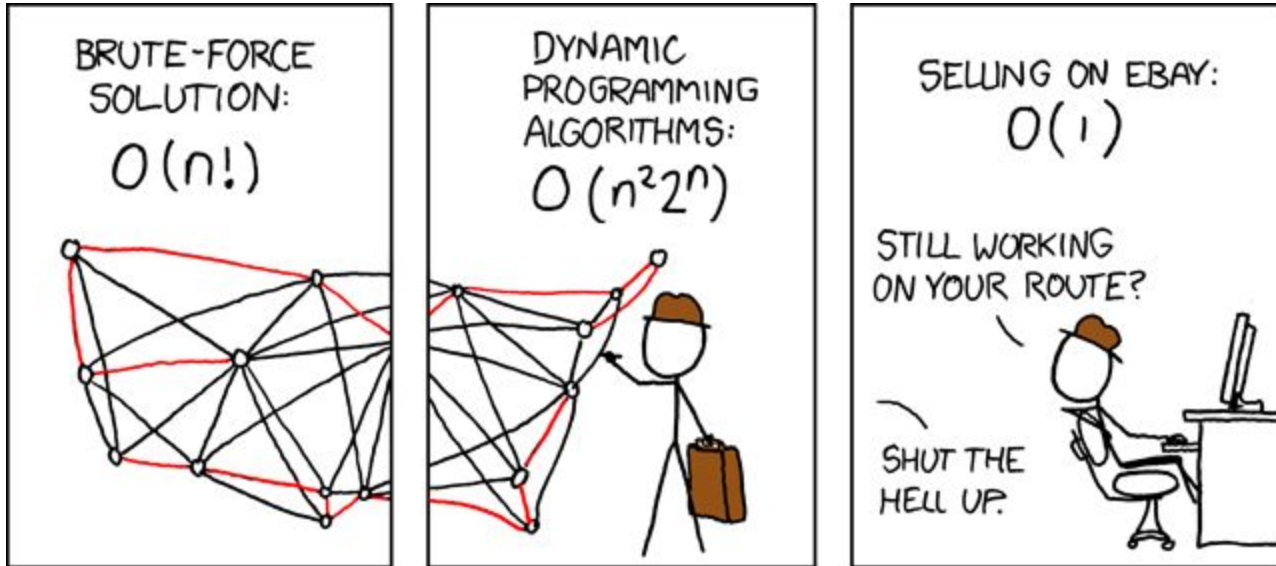
C'est donc cette méthode que nous allons développer dans la suite de ce cours.

Méthodes approchées

Une autre solution consiste à utiliser des méthodes approchées efficaces. Dans ce cas nous n'avons plus de problèmes aux limites, mais aucune garantie d'optimalité ne peut être avancée. L'algorithme basé sur l'arbre couvrant de poids minimal vu l'an passé en cours de graphe est une méthode de ce type et garantit simplement de ne pas dépasser d'un facteur 2 le coût de la solution optimale.

D'autres approches basées sur des algorithmes heuristiques (recuit simulé, recherche taboue ou algorithmes génétiques) sont également possibles, mais n'offrent que des approximations qu'il faut vérifier expérimentalement.

Traveling salesman problem by xkcd



Algorithme de Little

1. Calcul des regrets de la matrice et réduction
2. Calcul des chemins par évictions successives
3. Application des coupures (branch and bound)

X1	X2	X3	X4	X5	
--	11	1	7	9	X1
5	--	3	12	3	X2
7	1	--	9	13	X3
14	9	5	--	4	X4
3	12	7	1	--	X5

Little (calcul des regrets)

Cette étape consiste à évaluer, comme dans les problèmes de transport du chapitre précédent, les regrets associés aux lignes et aux colonnes.

$$R1=1+3+1+4+1=10$$

$$R2= 2+0+0+0+0=2$$

X1	X2	X3	X4	X5	
--	11	1	7	9	X1
5	--	3	12	3	X2
7	1	--	9	13	X3
14	9	5	--	4	X4
3	12	7	1	--	X5



X1	X2	X3	X4	X5	
--	10	0	6	8	X1
2	--	0	9	0	X2
6	0	--	8	12	X3
10	5	1	--	0	X4
2	11	6	0	--	X5

X1	X2	X3	X4	X5	
--	10	0	6	8	X1
0	--	0	9	0	X2
4	0	--	8	12	X3
8	5	1	--	0	X4
0	11	6	0	--	X5

On obtient par réduction les matrices de coût réduites et une première borne estimant le coût minimal : $C=R1+R2=10+2=12$

Little (éviction)

Le coût d'éviction d'une case de la matrice correspond à la somme minimale des cases différentes sur la même ligne et sur la même colonne. Elle représente le surcoût minimal engendré par le non choix d'une case. On choisit donc la case nulle (valeur 0) dont le coût d'éviction est maximal.

X1	X2	X3	X4	X5	
--	10	0(6)	6	8	X1
0	--	0(0)	9	0(0)	X2
4	0(9)	--	8	12	X3
8	5	1	--	0(1)	X4
0(0)	11	6	0(6)	--	X5

Little (éviction suite)

Suppression de la ligne et de la colonne correspondante

Rendre infini le coût de retour (on met un --).

Réduire la nouvelle matrice.

X1	X3	X4	X5	
--	0	6	8	X1
0	0	9	0	X2
8	1	--	0	X4
0	6	0	--	X5

X1	X3	X4	X5	
--	0	6	8	X1
0	--	9	0	X2
8	1	--	0	X4
0	6	0	--	X5

Little (itération pour trouver un chemin)

X1	X3	X4	X5	
--	0(7)	6	8	X1
0(0)	--	9	0(0)	X2
8	1	--	0(1)	X4
0(0)	6	0(6)	--	X5

X1	X4	X5	
--	9	0(9)	X2
8	--	0(8)	X4
0(8)	0(9)	--	X5

X1	X4	
8	--	X4
--	0	X5

Little (branch and bound)

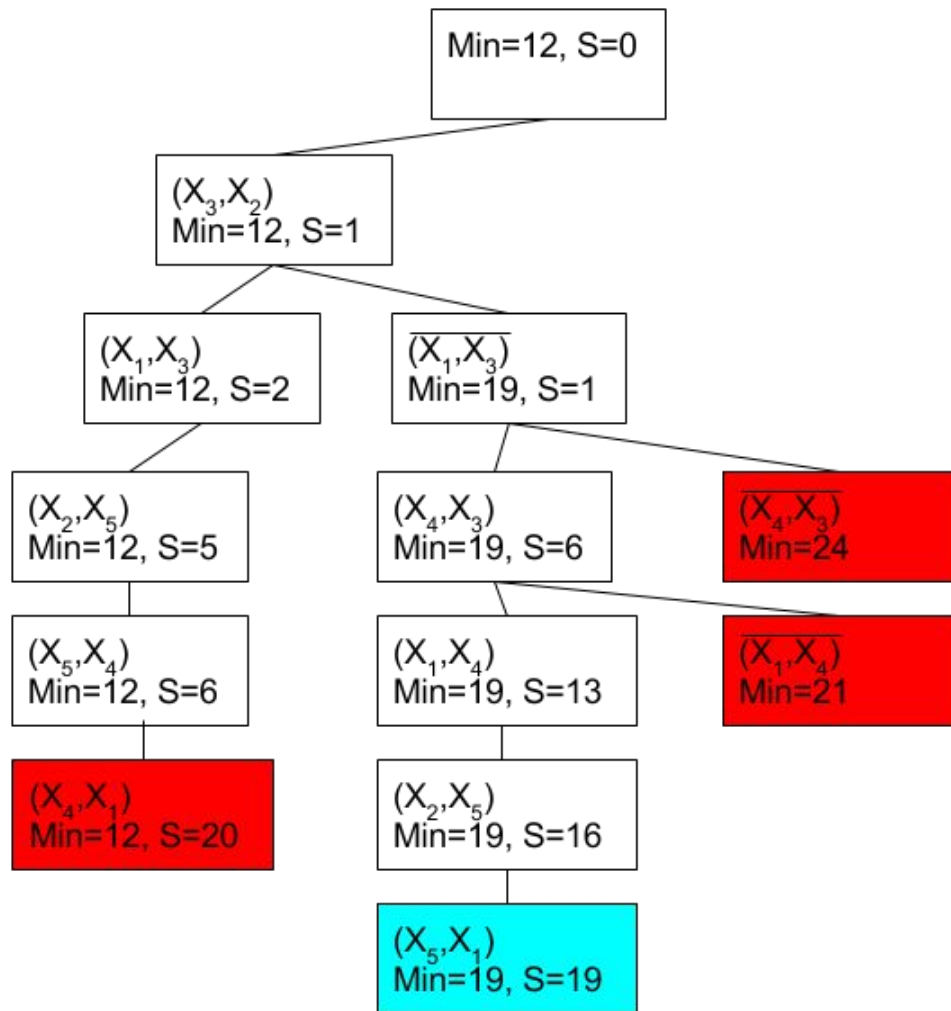
Construire l'arbre des solutions en suivant l'algorithme du branch and bound, dont la première solution, que l'on vient de calculer est la suivante :

$$(X3,X2) \rightarrow (X1,X3) \rightarrow (X2,X5) \rightarrow (X4,X1) \rightarrow (X5,X4)$$

$$\text{le coût } C = 12 \quad +0 \quad +0 \quad +0 \quad +8 \quad +0 \quad = 20$$

Couper toutes les solutions partielles > 20 et continuer la construction de l'arbre des solutions en répétant les étapes 1,2,3 (réduction, éviction, coupure)

Vous allez trouver une solution de coût $C=19$, optimale



Bibliowebographie

[Programmation dynamique par Tim Roughgarden](#)

[Introduction to intractability by Robert Sedgewick](#)