

Application #2

Problème du voyageur de commerce (TSP)

MTH6311

S. Le Digabel, École Polytechnique de Montréal

H2018

(v4)

Plan

1. Introduction
2. Formulations MIP
3. Heuristiques pour le TSP

1. Introduction

2. Formulations MIP

3. Heuristiques pour le TSP

Introduction

- ▶ Le problème du voyageur de commerce, ou TSP pour *Traveling-Salesman Problem*, consiste, pour un graphe donné, de déterminer un cycle hamiltonien dont la longueur est minimale.
- ▶ Pas juste des villes et des distances : le TSP peut se rencontrer dans d'autres contextes, et/ou comme sous-problème : problèmes de logistique, de transport, d'ordonnancement, etc. (voir [problème du père Noël](#)).
- ▶ Certains problèmes rencontrés dans l'industrie se modélisent sous la forme d'un TSP, comme l'optimisation de trajectoires de machines outils : comment percer plusieurs points sur une carte électronique le plus vite possible ?

Introduction (suite)

- ▶ Un des accomplissements majeurs du TSP est l'aide donnée au domaine *Mixed-Integer Programming* (MIP). Quasiment tout algorithme pour résoudre un MIP a d'abord été testé sur le TSP.
- ▶ La théorie de la complexité a été établie par W. Cook en 1971 en se basant sur l'exemple du TSP.
- ▶ Le TSP est \mathcal{NP} -difficile.
- ▶ Liens : [site du TSP](#) et [TSP interactif](#).

Notations

- ▶ $G = (V, E)$.
- ▶ Chaque sommet représente une ville. $|V| = n$.
- ▶ On considère que le graphe est complet : $|E| = n(n - 1)/2$.
- ▶ G est non-orienté mais les solutions doivent tenir compte d'une orientation.
- ▶ c_{ij} : coût (distance) de l'arc $(i, j) \in E$. On suppose que $c_{ij} = c_{ji}$ pour tout $(i, j) \in E$.
- ▶ On peut supposer aussi que les distances sont positives et respectent l'inégalité triangulaire

$$c_{ij} + c_{jk} \geq c_{ik} \text{ pour tous } i, j, k \in V .$$

- ▶ Pour un tour $T \subseteq E$, on note le coût du tour $c(T) = \sum_{e \in T} c(e)$.

Histoire non-exhaustive du TSP

- ▶ Premières références en 1832.
- ▶ 1859, W.R. Hamilton : énoncé avec un voyageur de commerce.
- ▶ 1949, J.B. Robinson, *“On the Hamiltonian game (a traveling-salesman problem)”*. Première référence sous sa forme moderne.
- ▶ Années 30-50 : popularisé dans la communauté mathématique par M. Flood et les chercheurs du RAND.
- ▶ 1954, RAND : G. Dantzig, R. Fulkerson, et S. Johnson, *“Solution of a large-scale traveling-salesman problem”*. Solution exacte pour les 49 capitales des états américains (introduction des coupes et du B&B).

Histoire non-exhaustive du TSP (suite)

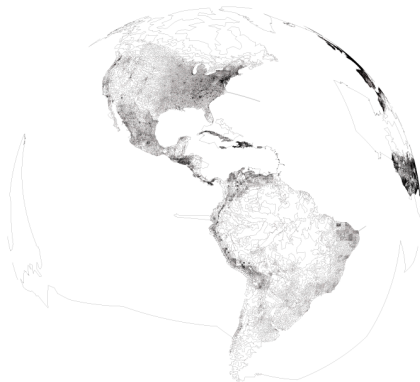
- ▶ 1962, M. Held et R.M. Karp : introduction d'heuristiques basées sur la programmation dynamique.
- ▶ 1973 : Heuristique de Lin and Kernighan.
- ▶ 1976 : Heuristique de Christofides.
- ▶ Années 80 – aujourd'hui : diverses heuristiques et métaheuristiques.

Tailles des instances résolues

| Année | Chercheurs | <i>n</i> |
|-------|---|----------|
| 1954 | Dantzig, Fulkerson, Johnson | 49 |
| 1971 | Held, Karp | 64 |
| 1975 | Camerini, Fratta, Maffioli | 67 |
| 1977 | Grötschel | 120 |
| 1980 | Crowder, Padberg | 318 |
| 1987 | Padberg, Rinaldi | 532 |
| 1987 | Grötschel, Holland | 666 |
| 1987 | Padberg, Rinaldi | 2,392 |
| 1994 | Applegate, Bixby, Chvátal, Cook | 7,397 |
| 1998 | Applegate, Bixby, Chvátal, Cook | 13,509 |
| 2001 | Applegate, Bixby, Chvátal, Cook | 15,112 |
| 2004 | Applegate, Bixby, Chvátal, Cook, Helsgaun | 24,978 |
| 2005 | Cook et al. | 33,810 |
| 2006 | Cook et al. | 85,900 |

Problème ouvert : le World TSP

- ▶ Instance de $n = 1,904,711$ villes.
- ▶ www.tsp.gatech.edu/world/index.html.
- ▶ Meilleure borne actuelle (2007) : 7,512,218,268.
- ▶ Meilleure solution actuelle (2011) : 7,515,778,188 : *gap* d'au plus 0.0474%.



1. Introduction

2. Formulations MIP

3. Heuristiques pour le TSP

Une première formulation MIP

On utilise deux variables binaires x_{ij} et x_{ji} pour chaque arête $(i, j) \in E$. La variable x_{ij} vaut 1 si on emprunte l'arête de i vers j , 0 sinon.

$$\min_x \sum_{(i,j) \in E} c_{ij}(x_{ij} + x_{ji})$$

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1 \quad \forall j \in V \quad (\text{entrer une seule fois dans chaque ville})$$

$$\sum_{\substack{j \in V \\ j \neq i}} x_{ij} = 1 \quad \forall i \in V \quad (\text{sortir une seule fois de chaque ville})$$

$$x_{ij} \text{ et } x_{ji} \in \{0, 1\} \quad \forall (i, j) \in E$$

Une première formulation MIP

On utilise deux variables binaires x_{ij} et x_{ji} pour chaque arête $(i, j) \in E$. La variable x_{ij} vaut 1 si on emprunte l'arête de i vers j , 0 sinon.

$$\min_x \sum_{(i,j) \in E} c_{ij}(x_{ij} + x_{ji})$$

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1 \quad \forall j \in V \quad (\text{entrer une seule fois dans chaque ville})$$

$$\sum_{\substack{j \in V \\ j \neq i}} x_{ij} = 1 \quad \forall i \in V \quad (\text{sortir une seule fois de chaque ville})$$

$$x_{v_1 v_2} + x_{v_2 v_3} + \dots + x_{v_t v_1} \leq t - 1 \quad \forall S = \{v_1, v_2, \dots, v_t\} \subset V$$

(élimination des sous-tours)

$$x_{ij} \text{ et } x_{ji} \in \{0, 1\} \quad \forall (i, j) \in E$$

Une première formulation MIP (suite)

- ▶ Contraintes d'élimination des sous-tours :
 - ▶ Imposent une tournée reliant tous les sommets de G .
 - ▶ Pour $t = 2$, revient à $x_{ij} + x_{ji} \leq 1$.
 - ▶ Pour $t = 3$, $x_{ij} + x_{jk} + x_{ki} \leq 2$ ou bien $x_{ik} + x_{kj} + x_{ji} \leq 2$ (élimine les deux tours dans un triangle).
 - ▶ Au pire 2^n contraintes de bris de sous-tours. A-t-on besoin de toutes ces contraintes? Peut-on les générer itérativement quand un sous-tour apparaît dans une tournée?
- ▶ Résolution par la technique de séparation et évaluation (*Branch and Bound*) ou techniques plus spécialisées.

Une deuxième formulation MIP (flots et étapes)

On utilise un indice supplémentaire $s \in \{1, 2, \dots, n\}$ (l'étape) : la variable $x_{ij,s}$ vaut 1 si on va de i à j à l'étape s , 0 sinon :

$$\min_x \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{s=1}^n c_{ij} x_{ij,s}$$

$$\sum_{i=1}^n x_{ij,s} - \sum_{i=1}^n x_{ji,s \% n + 1} = 0 \quad \forall j \text{ et } s \in \{1, 2, \dots, n\}$$

(flot entrant = flot sortant)

$$\sum_{j=1}^n \sum_{s=1}^n x_{ij,s} = 1 \quad \forall i \in \{1, 2, \dots, n\} \text{ (passer une fois par chaque ville)}$$

$$x_{ij,s} \in \{0, 1\} \quad \forall i, j, s \in \{1, 2, \dots, n\}$$

On n'a plus besoin des contraintes d'élimination des sous-tours, mais cette formulation requiert n^3 variables : 50 villes donnent 125,000 variables !

1. Introduction

2. Formulations MIP

3. Heuristiques pour le TSP

Introduction

- ▶ On voit ici deux types d'heuristiques :
 - ▶ Celles qui construisent une tournée en ne partant de rien (*from scratch*).
 - ▶ Celles, plus efficaces, qui améliorent une tournée déjà existante.
- ▶ Les heuristiques vues ici sont dédiées au TSP et ne sont donc pas des métaheuristiques. Cependant plusieurs notions, comme les voisinages, peuvent être utilisées pour définir des métaheuristiques.
- ▶ Ensemble d'instances : [TSPLIB](#).

Voisin le plus proche

Nearest Neighbour (NN)

[1] Choisir un sommet $v_1 \in V$
Poser $k \leftarrow 1$

[2] Tant que $k < n$
 | $k \leftarrow k + 1$
 | choisir v_k dans $V \setminus \{v_1, v_2, \dots, v_{k-1}\}$
 | qui minimise c_{v_{k-1}, v_k}

Retourner le tour (v_1, v_2, \dots, v_n)

Voisin le plus proche (suite)

- ▶ Facile à implémenter. Complexité en $\mathcal{O}(n^2)$.
- ▶ Sur les problèmes de la **TSPLIB**, en moyenne, NN donne des tours de coût 1.26 fois plus élevé que la valeur optimale, noté $NN/OPT = 1.26$.
- ▶ Pire comportement connu : $NN/OPT \in \Theta\left(\frac{\log n}{3 \log \log n}\right)$ (environ 3 pour $n = 100$, 4 pour $n = 10,000$, etc.)
- ▶ **Borne garantie sur la solution optimale** (Rosenkrantz, Stearns, Lewis, 1977) : Si les distances sont positives et respectent l'inégalité triangulaire, alors

$$NN/OPT \leq \frac{1}{2} \lceil \log_2 n \rceil + \frac{1}{2}$$

(4.5 pour $n = 100$, 7.5 pour $n = 10,000$, etc.)

Méthodes d'insertion

On commence avec deux noeuds et on ajoute les autres noeuds un à un. Contrairement à NN, à chaque étape, on a un tour.

Insertion Method (IM)

[1] Choisir v_1 et $v_2 \in V$ et poser $T = (v_1, v_2)$
(typiquement on maximise $c_{v_1 v_2}$)

Poser $k \leftarrow 2$

[2] Tant que $k < n$

$k \leftarrow k + 1$

 choisir v_k dans $V \setminus T$

 insérer v_k dans T

Retourner T

Méthodes d'insertion (suite)

Pour un noeud v_k hors du tour T :

- ▶ La quantité $d_T(v_k) = \min_{v \in T} c_{v_k, v}$ représente la distance au tour.
- ▶ Un sommet $v_T(v_k) \in \arg \min_{v \in T} c_{v_k, v}$ est le sommet du tour le plus proche de v_k .

Méthodes d'insertion (suite)

Il y a plusieurs stratégies pour choisir et insérer les noeuds, toutes en $\mathcal{O}(n^2)$, ce qui donne une complexité en $\mathcal{O}(n^3)$ pour la méthode au complet.

- ▶ **Nearest insertion (IM-N)** : On insère le noeud le plus proche du tour : $\min_{v_k \in V \setminus T} d_T(v_k)$. Dans T , on place v_k juste après

$v_T(v_k)$.

- ▶ **Cheapest insertion (IM-C)** : On choisit le noeud dont l'insertion dans T sera la moins coûteuse :

$\min_{v_k \in V \setminus T} \min_{\substack{v_1, v_2 \in T \\ \text{consécutifs}}} c_{v_1, v_k} + c_{v_k, v_2}$. On place v_k dans T entre v_1 et v_2 .

- ▶ **Farthest insertion (IM-F)** : On choisit le noeud le plus loin du tour : $\max_{v_k \in V \setminus T} d_T(v_k)$. Dans T , on place v_k juste après

$v_T(v_k)$.

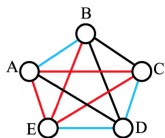
Méthodes d'insertion (suite)

- ▶ Sur **TSPLIB**, en moyenne, $IM-F/OPT = 1.16$. C'est la plus efficace des trois méthodes en pratique.
- ▶ **Borne garantie sur la solution optimale** (Rosenkrantz, Stearns, Lewis, 1977) : Si les distances sont positives et respectent l'inégalité triangulaire, alors
 - ▶ $IM-N/OPT \leq 2$.
 - ▶ $IM-C/OPT \leq 2$.
 - ▶ $IM-F/OPT \leq \lceil \log_2 n \rceil + 1$
(8 pour $n = 100$, 15 pour $n = 10,000$, etc.)
- ▶ Jusqu'à présent, aucun exemple connu pour lequel $IM-X/OPT > 4$.

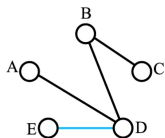
Arbre de recouvrement (MST)

- ▶ Trouver un **arbre de recouvrement** T de G de poids minimum. Pour cela, utiliser **l'algorithme de Kruskal**, par exemple, en $\mathcal{O}(n^2 \log n)$.
- ▶ Déterminer un cycle qui passe exactement deux fois par chaque arête de l'arbre (et nulle part ailleurs).
- ▶ Emprunter des raccourcis pour éviter de passer plusieurs fois par un même sommet.
- ▶ Borne sur la solution optimale : $\text{MST}/\text{OPT} \leq 2$.

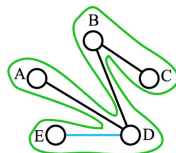
Arbre de recouvrement : illustration



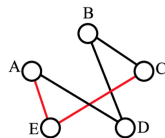
Les arêtes noires sont de longueur 1
 Les bleues sont de longueur 2
 Les rouges sont de longueur 3



Arbre de coût minimum.
 Son coût est de 5



Cycle (A,D,B,C,B,D,E,D,A)
 de coût 10



Cycle raccourci (A,D,B,C,E,A)
 de coût 9

Heuristique de Christofides (1976)

L'heuristique comporte quatre étapes pour une complexité en $\mathcal{O}(n^3)$:

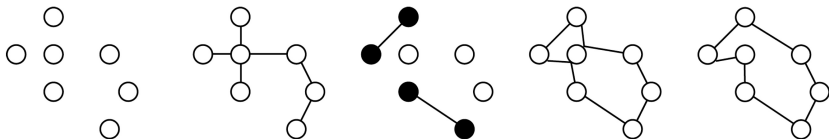
1. Trouver un arbre de recouvrement T de G de poids minimum. Utiliser l'**algorithme de Kruskal** en $\mathcal{O}(n^2 \log n)$.
2. Soient W l'ensemble des noeuds de degré impair dans T et $G(W)$ le sous-graphe de G induit par W .

L'étape 2 consiste à trouver un **couplage parfait** M de $G(W)$ de coût minimal. Pour cela, utiliser l'**algorithme d'Edmonds**, par exemple, en $\mathcal{O}(n^3)$.

Heuristique de Christofides (suite)

3. On construit le graphe J en unissant les arêtes de T et le couplage M . Si une arête est à la fois dans T et dans M , alors on la dédouble dans J . J est un graphe connexe dont les sommets sont ceux de V et ont des degrés pairs. Ce graphe définit donc un tour, et si tous les degrés sont de 2, alors le tour est hamiltonien et on ne fait pas l'étape 4.
4. *Raccourcis* : Pour tout sommet v de degré > 2 , considérer deux arêtes (u, v) et (v, w) consécutives dans le tour de J et les remplacer dans J par l'arête (u, w) .

Heuristique de Christofides : illustration



Heuristique de Christofides (suite)

- ▶ Sur **TSPLIB** :
 - ▶ En moyenne, $\text{CHR}/\text{OPT} = 1.14$.
 - ▶ Si à l'étape 4 les meilleurs raccourcis possibles sont choisis, alors, en moyenne, $\text{CHR}/\text{OPT} = 1.09$.
- ▶ **Borne garantie sur la solution optimale** : Si les distances sont positives et respectent l'inégalité triangulaire, alors $\text{CHR}/\text{OPT} \leq 1.5$ (preuve avec les algorithmes d'approximation).

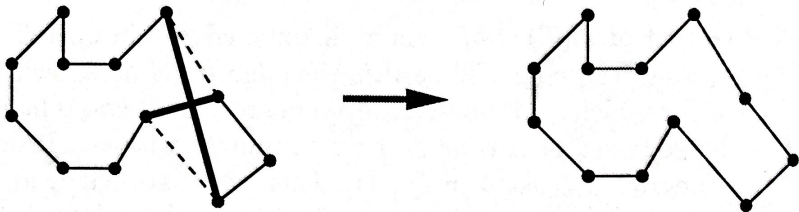
Amélioration de tour : méthodes k -opt

- ▶ Considèrent un tour T déjà existant.
- ▶ Avec le 2-opt, pour chaque paire d'arêtes non-adjacentes dans T , en supprimant ces arêtes, on crée deux chemins qu'on peut recombinaisonner en un nouveau tour T' .
- ▶ Si $c(T') < c(T)$, alors on remplace T par T' et on continue.
- ▶ Si $c(T') \geq c(T)$ pour toute paire d'arêtes non-adjacentes dans T , alors T est dit **2-optimal** et l'algorithme se termine.

Algorithme 2-opt : complexité

- ▶ Vérifier qu'un tour est 2-optimal se fait en temps polynomial : $\mathcal{O}(n^2)$.
- ▶ Mais trouver un tour 2-optimal ne se fait pas en temps polynomial.
- ▶ Papadimitriou et Steiglitz (1977) : si on fait de mauvais choix, on peut prendre un nombre exponentiel d'échanges avant de trouver un tour 2-optimal.

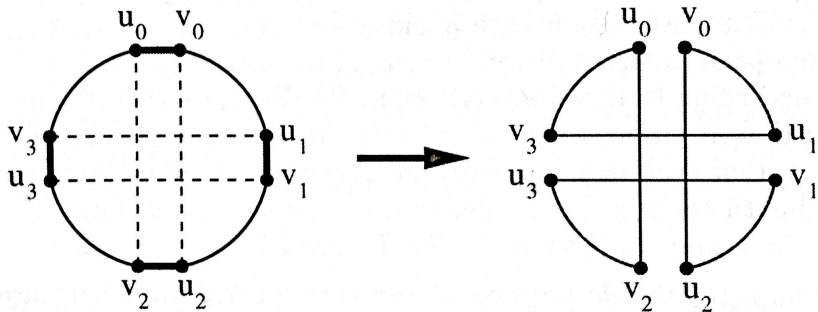
2-opt : exemple



Généralisation à k -opt

- ▶ On fait la même chose que 2-opt, mais cette fois il faut considérer des sous-ensembles d'arêtes de T de cardinalité k .
- ▶ Une fois ces arêtes supprimées, il faut être capable de reformer un nouveau tour.
- ▶ Ceci est de plus en plus difficile quand k grandit.
- ▶ En pratique, on considère rarement $k > 3$.
- ▶ Sur [TSPLIB](#) : 2-OPT/OPT= 1.06 et 3-OPT/OPT= 1.04.

4-opt : exemple



Heuristique de Lin-Kernighan (1973)

- ▶ Méthode d'amélioration de tour.
- ▶ Généralisation de 2-opt et 3-opt : méthode adaptative qui décide à chaque étape du nombre d'arêtes à échanger.
- ▶ Quand une amélioration est obtenue, elle n'est pas forcément utilisée tout de suite.
- ▶ Algorithme très technique, donc non présenté ici.
- ▶ Lin-Kernighan + composante 4-opt : *Chained Lin-Kernighan*.
- ▶ Sur [TSPLIB](#) : $LK/OPT = 1.02$ et $CLK/OPT = 1.01$.