BALASSOUPRAMANIEN MADAVAN
CHEVILY PIERRE
POMMIER MELVYN

ING1 APP - LSI3

RAPPORT

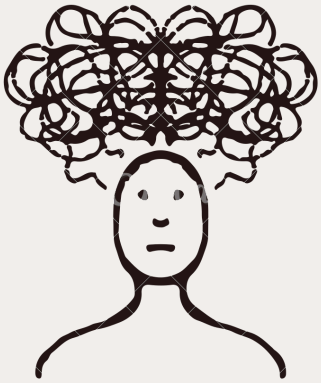
Le problème du voyageur de commerce

OPTIMISATION ET COMPLEXITÉ
CHARRAD TASNIM

2024 - 2025

SOMMAIRE

Définition du problème

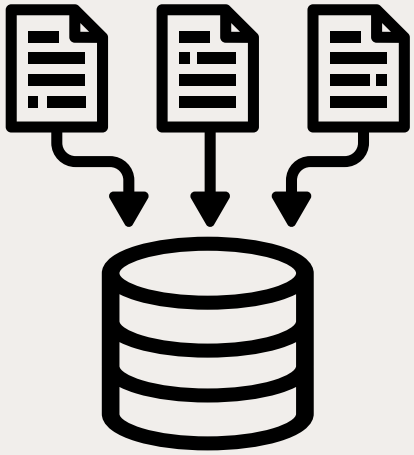


LE PROBLÈME

Etant donné une liste de villes et les distances entre chaque paire de villes, trouver le **plus court chemin** pour passer par toutes les villes de la liste, une **seule et unique fois**.

DONNÉES D'ENTRÉE

- Un ensemble de villes $V = \{v_1, v_2, ..., v_n\}$
- Une matrice de distances $D = [d_{ij}]$
  où d_{ij} est la distance entre les villes v_i et v_j



OBJECTIF

On souhaite trouver une permutation de **π**, de l'ensemble des **villes** qui **minimise** la distance totale du tour. Soit :

$$\min_{\pi} \left(\sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \right)$$

Revue de la littérature

Le problème du voyageur de commerce est un problème classique d'optimisation combinatoire. Il s'agit de trouver le plus court chemin qui permet de visiter une fois chaque ville d'un ensemble donné, puis de revenir au point de départ.

C'est un problème NP-difficile, ce qui signifie qu'il devient très complexe à résoudre lorsque le nombre de villes augmente. Il est pourtant fondamental car il est à la base de nombreuses applications dans la logistique, la planification, ou encore la robotique.

PRINCIPALES APPROCHES :

- Méthodes exactes :

Utilisent des outils **mathématiques** (programmation linéaire, branch and bound) pour garantir la meilleure solution. Trop **lentes** pour les grands cas.

- Heuristiques :

Fournissent des solutions assez bonnes **rapidement**. Exemples : plus proche voisin, insertion, 2-opt.

- Métaheuristiques :

Algorithmes inspirés de la nature ou de l'évolution (recuit simulé, génétiques, fourmis, recherche tabou). Très **efficaces** pour les grands problèmes.

POURQUOI ON L'ÉTUDIE ?

- Parce qu'il est **simple** à **comprendre**, mais très **difficile** à **résoudre**.
- Parce qu'on le retrouve **partout** dans la vraie vie : livraison de colis, circuits imprimés, organisation de tournées, etc.
- Parce qu'il existe de nombreuses variantes plus proches des **cas réels** (avec fenêtres de temps, plusieurs véhicules, contraintes de capacité...).

Formulation mathématique

Le Problème du Voyageur de Commerce (TSP) peut être formulé comme un programme linéaire en nombres entiers. Une des formulations les plus connues est celle proposée par Dantzig, Fulkerson et Johnson (DFJ), complétée ici par les contraintes de Miller-Tucker-Zemlin (MTZ) pour éliminer les sous-tours.

VARIABLES

- $x_{ij} \in \{0, 1\}$:
  vaut 1 si le trajet va de la ville i à la ville j, 0 sinon.
- $u_i \in \mathbb{R}$:
  variable auxiliaire utilisée pour éliminer les sous-tours (utile uniquement pour les villes 2 à n).

FONCTION OBJECTIF

Minimiser la distance totale parcourue :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot x_{ij}$$

où d_{ij} est la distance (ou le coût) entre la ville i et la ville j.

CONTRAINTES

01

$$\sum_{j=1}^{n} x_{ij} = 1$$

$$\forall i \in \{1, \ldots, n\}$$

Chaque ville est quittée une seule fois

02

$$\sum_{i=1}^{n} x_{ij} = 1$$

$$\forall j \in \{1, \ldots, n\}$$

Chaque ville est visitée une seule fois

03

$$u_i - u_j + n \cdot x_{ij} \leq n - 1$$
$$\forall i \neq j,\ i, j \in \{2, \ldots, n\}$$

$$1 \leq u_i \leq n - 1$$
$$\forall i \in \{2, \ldots, n\}$$

Élimination des sous-tours (MTZ)

Cette formulation garantit que la solution obtenue constitue un seul cycle passant par toutes les villes, sans création de circuits partiels (sous-tours).

Résolution du programme

Notre projet propose une résolution du problème du voyageur de commerce (TSP) via une approche exacte basée sur la programmation linéaire en nombres entiers, en utilisant :

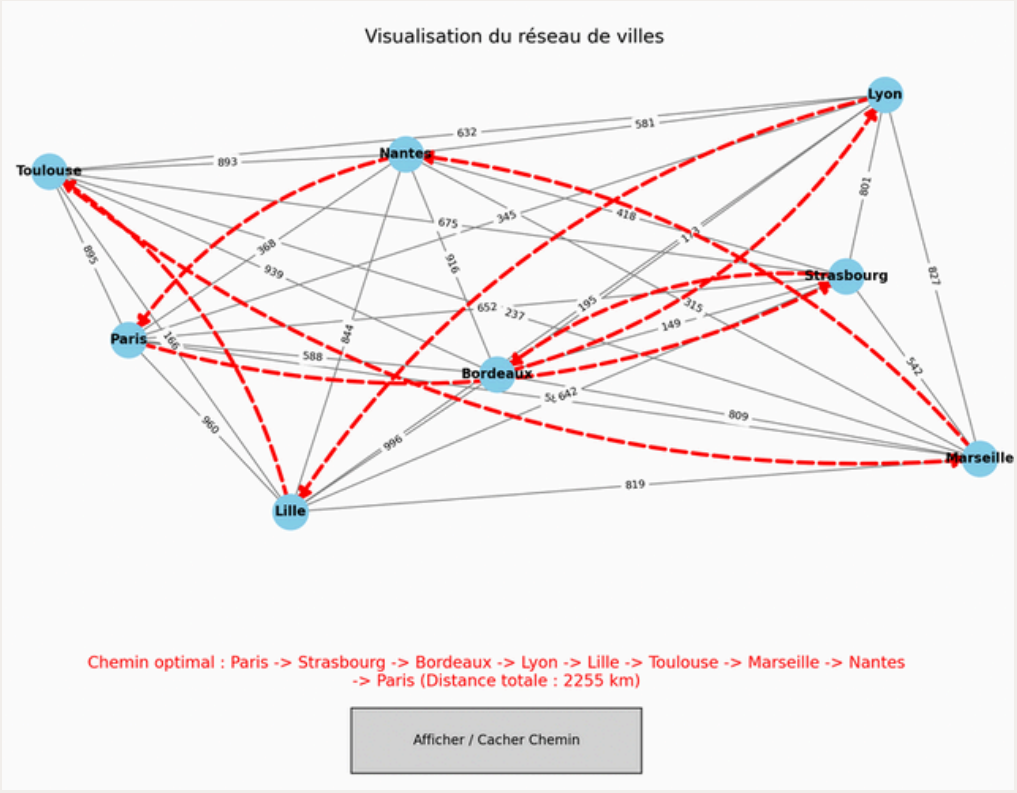
- Le solveur PuLP (avec CBC en backend).
- La formulation MTZ (Miller–Tucker–Zemlin) pour éviter les sous-tours.
- La génération automatique d'une matrice de distances aléatoires pour simuler des instances réalistes.

Une fois la solution trouvée, elle est visualisée de manière dynamique grâce à la bibliothèque NetworkX et Matplotlib.

FONCTIONNALITÉS PRINCIPALES DÉVELOPPÉES

- Génération automatique de données (villes et distances).
- Construction du modèle mathématique du TSP (variables, contraintes MTZ).
- Résolution optimale du problème via le solveur CBC.
- Visualisation interactive du graphe des villes :
  - Affichage initial du réseau complet.
  - Affichage optionnel du chemin optimal avec :
    - Arêtes rouges en pointillé et flèches directionnelles.
    - Distance totale et ordre de passage affichés en bas.
    - Bouton pour afficher/cacher le chemin optimal dynamiquement.

Résultats obtenus



- Temps de résolution : < 2sec pour 30 villes.
- Solution optimale trouvée.
- Visualisation dynamique du chemin optimal possible.

REMARQUES & AMÉLIORATIONS POSSIBLES

- Le solveur CBC reste efficace jusqu'à environ 30 villes ; au-delà, d'autres méthodes heuristiques seraient plus adaptées.
- Des options d'amélioration pourraient être :
  - Intégrer un système de sélection de villes personnalisées.
  - Tester différentes formulations (DFJ vs MTZ).
  - Optimiser la visualisation pour de très grands graphes.