BALASSOUPRAMANIEN MADAVAN
CHEVILY PIERRE
POMMIER MELVYN

ING1 APP - LSI3

PRÉSENTATION

Le problème du voyageur de commerce

OPTIMISATION ET COMPLEXITÉ
CHARRAD TASNIM

2024 - 2025

SOMMAIRE

DÉFINITION DU PROBLÈME

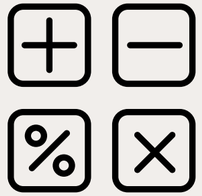
OBJECTIF

Trouver le **plus court circuit** visitant toutes les villes **une seule fois**

DONNÉES D'ENTRÉE

- Un ensemble de **villes**
- Une matrice de **distances**

$$D = [d_{ij}]$$

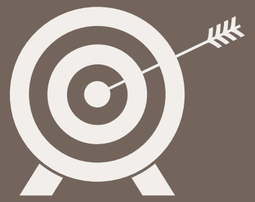


FORME DU RÉSULTAT

Une permutation **π** des villes **minimisant** la distance totale.

REVUE DE LA LITTÉRATURE

Principales approches



EXACTES

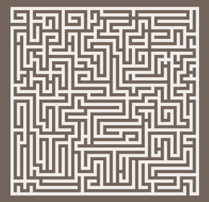
PLNE, Branch & Bound

Solutions **optimales**, mais **lentes**

HEURISTIQUES

Plus proche voisin, 2-opt

Résultats **rapides** et **corrects**



MÉTAHEURISTIQUES

Recuit simulé, génétiques, fourmis

Solutions **efficaces** pour les **grandes instances**

REVUE DE LA LITTÉRATURE

Pourquoi ce problème ?

PRÉSENT DANS LA VIE DE TOUS LES JOURS

Problème récurrent en

- **Logistique** (livraisons)
- **Bio-informatique** (ADN)
- **Circuits imprimés**
- **Robotique** (trajectoires)

SIMPLE À COMPRENDRE, PAS À RÉSOUDRE

Bien qu'il s'agisse d'un problème que l'on rencontre souvent, **personne** n'a, à ce jour, réussi à le résoudre, malgré de **nombreuses tentatives**.

D'AUTRES CONTRAINTES DANS LA VRAIE VIE

- Limites de **temps**
- **Capacité** des véhicules (livraison)
- Distance **A/R différentes**
- Ordre de **priorité**

FORMULATION MATHÉMATIQUE

Variables

$$x_{ij} \in \{0, 1\}$$

MATRICE

Vaut 1 si on va de **i à j**,
0 sinon

$$u_i \in \mathbb{R}$$

SOUS-TOURS

Variable pour **éliminer** les **sous-tours**

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

FONCTION OBJECTIF

On doit **minimiser** d_{ij} la distance entre les villes **i** et **j**

FORMULATION MATHÉMATIQUE

Contraintes

$$\sum_{j=1}^{n} x_{ij} = 1$$
$$\forall i \in \{1, \dots, n\}$$

DÉPART

On **quitte** chaque ville **une seule fois**

$$\sum_{i=1}^{n} x_{ij} = 1$$
$$\forall j \in \{1, \dots, n\}$$

VISITE

On **visite** chaque ville **une seule fois**

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

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

SOUS-TOURS

On **élimine** les sous-tours grâce au **MTZ**

RÉSOLUTION DU PROGRAMME

Utilisation de **PuLP** avec solveur **CBC**.

Génération **automatique** d'une matrice de distances réalistes

Modélisation via **formulation MTZ** (Miller–Tucker–Zemlin) pour éviter les **sous-tours**

Résolution rapide : moins de 2 sec pour 30 villes

```
2025-04-27 19:58:26 | SUCCESS  | manage_data.generate_data:generate_sample_distance_matrix:39 - Matrice de distances générée avec 8 villes.
2025-04-27 19:58:26 | SUCCESS  | manage_data.graph_plotter:build_graph:28 - Graphe construit avec 8 nœuds et 28 arêtes.
2025-04-27 19:58:26 | INFO     | solver.tsp_solver:create_model:53 - Modèle TSP construit avec succès.
2025-04-27 19:58:26 | INFO     | __main__:main:37 - Modèle construit, résolution en cours...
2025-04-27 19:58:26 | SUCCESS  | solver.tsp_solver:solve_model:77 - Résolution optimale obtenue.
2025-04-27 19:58:26 | INFO     | solver.tsp_solver:extract_solution:98 - Arêtes actives sélectionnées : [('Paris', 'Strasbourg'), ('Lyon', 'Lille
'), ('Nantes', 'Paris'), ('Bordeaux', 'Lyon'), ('Strasbourg', 'Bordeaux'), ('Lille', 'Toulouse')]
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 0: Paris -> Strasbourg
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 1: Strasbourg -> Bordeaux
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 2: Bordeaux -> Lyon
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 3: Lyon -> Lille
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 4: Lille -> Toulouse
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 5: Toulouse -> Marseille
2025-04-27 19:58:26 | DEBUG    | solver.tsp_solver:extract_solution:116 - Étape 6: Marseille -> Nantes
2025-04-27 19:58:26 | SUCCESS  | solver.tsp_solver:display_solution:147 -
======== SOLUTION TSP ========
Ordre optimal : Paris -> Strasbourg -> Bordeaux -> Lyon -> Lille -> Toulouse -> Marseille -> Nantes -> Paris
Distance totale estimée : 2255.00 km
```

FONCTIONNALITÉS

Génération aléatoire de villes et distances.

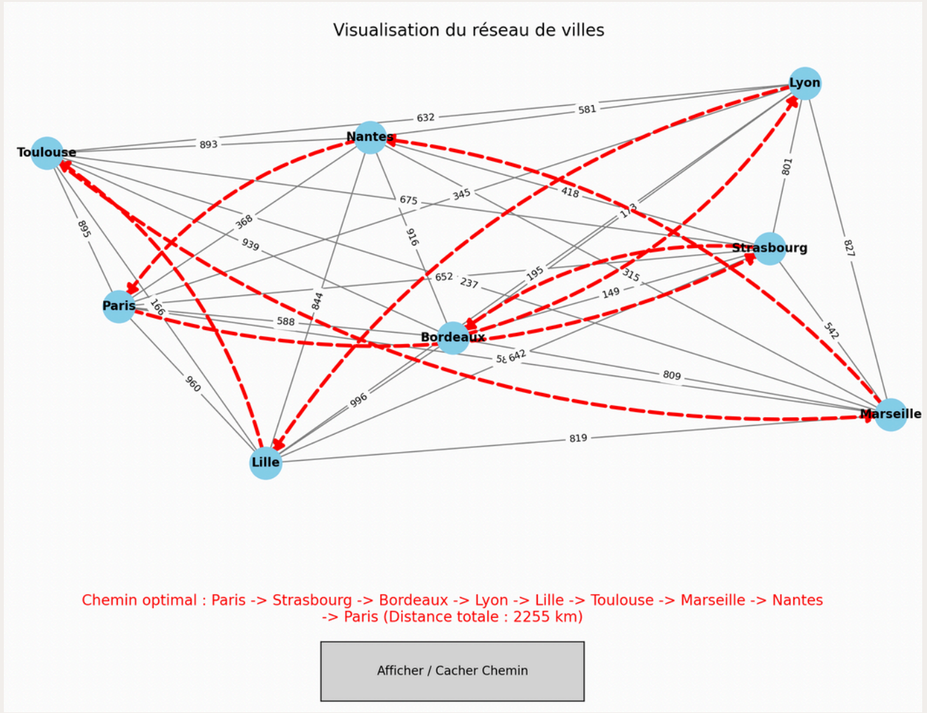
Construction du modèle mathématique : variables, contraintes.

Résolution optimale via CBC.

Visualisation interactive :

- Graphe complet affiché.
- Bouton "Afficher / Cacher Chemin optimal" :
  - Arêtes rouges pointillées avec flèches directionnelles.
  - Affichage du chemin optimal et de la distance totale.

RÉSULTATS OBTENUS



REMARQUES ET AMÉLIORATIONS POSSIBLES

CBC efficace jusqu'à ~40 villes.

Pour plus de 40 villes :

- Utiliser des heuristiques (2-opt, insertion, etc.).
- Tester d'autres formulations du TSP (DFJ au lieu de MTZ).
- Améliorer la visualisation pour très gros graphes (ex : clusterisation)