REMERCIEMENTS

TABLE DES MATIÈRES

1 Introduction **1**

2 Traveling Salesman Problem **2**

2.1 Algorithmes pour le résoudre 2

2.1.1 Nearest Neighbors 2

2.1.2 Algorithme 2-opt 3

2.1.3 Algorithme 3 4

2.1.4 Algorithme 4 4

3 Résultats **5**

4 Conclusion **6**

Bibliography **7**

GLOSSAIRE

Glossaire

CHAPITRE 1

INTRODUCTION

CHAPITRE 2
TRAVELING SALESMAN PROBLEM

Le problème du voyageur de commerce (TSP, par son nom en anglais) [1, 2] est un problème très connu et étudié dans le domaine de l'informatique et l'optimisation. Il cherche à répondre la question : « étant donné n villes et leurs distances, quel est le chemin le plus court qu'un voyageur peut faire pour passer pour tous les villes une seule fois eet revenir au point de départ ».

Le TSP comme problème est connu au moins depuis le 19ème siècle [1], mais sa formalisation mathématique généralisé apparaît dans les années 1930 [1] est popularisé avec ce nom par la RAND corporation en 1949 [1]. À partir de cette publication, le problème a devenu très populaire dû à la simplicité de son énoncé en comparaison a son résolution. Il est utilisé comme référence pour évaluer l'efficience des algorithmes.

Le TSP a diverses applications dans plusieurs domaines comme la logistique, séquençage d'ADN, manufacture, et autres. Chaque probleme composé d'un ensemble que doit être rempli dans un ordre particulier pourrait être pensé comme le problème TSP, où on cherche un chemin moins coûteux. Le problème peut être résolu exactement si tous les chemins entre villes sont analyses et comparés, mais la quantité de chemins a vérifier augmente rapidement. Pour ce raison, des algorithmes sont proposés pour trouver des solutions approximatives que peuvent être optimales ou pas, mais qui résolut le problème rapidement. Dans ce travail, seulement la version symétrique de l'algorithme sera analyse. Cela veut dire que les distances consideres entre villes est égales dans les deux sens. Les villes sont aussi considérés toutes interconnectés, donc il existera toujours une connection directe entre une ville et l'autre.

2.1 Algorithmes pour le résoudre

Le problème du voyageur de commerce est peut-être le plus étudié dans la théorie de la complexité, donc plusieurs des algorithmes ont été proposés pour trouver une solution optimale. Ici, quatre algorithmes à analyser sont présentés.

2.1.1 Nearest Neighbors

Le voisin plus proche [3] est un

Algorithme 2.1 : Algorithme pour les voisins plus proches

```
Entrées : Matrice de distance (distance[][]), Liste de villes (villes[])
Output : Ordre de visite (ordre_visite) et distance totale
         (distance_totale)

actuelle ← indice ville départ
visitee[] ← liste de booleans False pour chaque ville
ordre_visite[0] ← actuelle
visitee[actuelle] ← True
distance_totale ← 0

tant que il reste de villes à visiter faire
    meilleur_distance ← ∞
    meilleur_voisin ← None

    pour chaque voisin v et sa distance d de la ville actuelle faire
        si voisin pas visité et distance < meilleur_ distance alors
            meilleur_distance ← distance[actuelle][v]
            meilleur_voisin ← v

    si il n'y a pas de meilleur voisin alors
        break

    actuelle ← indice du meilleur voisin
    visitee[actuelle] ← True
    ordre_visite← ajouter ville actuelle
    distance_totale ← distance_totale + meilleur_distance

pour chaque ville visitée v en ordre inverse faire
    si distance avec ville de départ > 0 alors
        ordre_visite← ajouter ville de départ
        distance_totale ← distance_totale + distance[v][depart]
        actuelle ← indice du ville de départ
    sinon
        ordre_visite← ajouter ville antérieure
        distance_totale ← distance_totale + distance[v][v-1]
        actuelle ← indice du ville de antérieure

retourner ordre_visite, distance_totale
```

2.1.2 Algorithme 2-opt

L'algorithme 2-opt [4]

Algorithme 2.2 : 2-opt Optimization

Entrées : Initial tour from Nearest Neighbor
Output : Improved tour and total distance

improved ← `true`
tour ← visit_order excluding last city
best_distance ← total distance of tour + return to start

tant que *improved* **faire**
    improved ← `false`

    pour $i \leftarrow 1$ **à** $n-2$ **faire**
        pour $j \leftarrow i+1$ **à** $n-1$ **faire**
            new_tour ← tour with segment $[i:j]$ reversed
            new_distance ← distance of new_tour + return to start
            si *new_distance* $<$ *best_distance* **alors**
                tour ← new_tour
                best_distance ← new_distance
                improved ← `true`
                break

        si *improved* **alors**
            break

visit_order ← tour + [start city]
total_distance ← best_distance

retourner visit_order

2.1.3 Algorithme 3

2.1.4 Algorithme 4

CHAPITRE 3

RÉSULTATS

CHAPITRE 4

CONCLUSION

BIBLIOGRAPHIE

[1] Donald Davendra. *Traveling Salesman Problem, Theory and Applications.* InTech, Janeza Trdine 9, 51000 Rijeka, Croatia, first edition, 2010.

[2] Padberg Manfred Hoffman, Karla and Giovanni Rinaldi. *Traveling Salesman Problem (TSP)Traveling salesman problem*, pages 849–853. Springer US, New York, NY, 2001.

[3] Cor A.J. Hurkens and Gerhard J. Woeginger. On the nearest neighbor rule for the traveling salesman problem. *Operations Research Letters*, 32(1) :1–4, 2004.

[4] Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the tsp. *Algorithmica*, 68(1) :190–264, 2014.