Voyageur de Commerce

Mohamed amine KASMI

22 December 2023

1 Introduction

Le problème du voyageur de commerce (TSP) constitue l'un des défis les plus emblématiques en optimisation combinatoire. Imaginons un commerçant devant visiter un ensemble de villes et cherchant à minimiser la distance totale parcourue tout en passant par chaque ville exactement une fois avant de retourner à son point de départ. Bien que cette situation puisse sembler simple à première vue, elle donne lieu à un problème complexe et d'une grande importance pratique.

Le TSP possède des applications dans divers domaines tels que la logistique, la planification des itinéraires, la fabrication et même la conception de circuits électroniques. En raison de sa nature NP-complète, la recherche de la solution optimale devient rapidement difficile à mesure que le nombre de villes augmente. De ce fait, le TSP attire l'attention de chercheurs en informatique, en mathématiques et en ingénierie, qui travaillent sans relâche pour développer des algorithmes efficaces afin de résoudre ce problème, contribuant ainsi à résoudre des défis concrets liés à la gestion des ressources et à l'optimisation des trajets. Cette introduction souligne l'importance du TSP en tant que défi computationnel et pose les bases de la quête pour des solutions innovantes à ce problème complexe.

2 Recherche avec métaheuristique

Pour la recherche avec métaheuristique, j'ai utilisé le langage Python et j'ai implémenté plusieurs fonctions, telles que `generer_sol_ini(n)` qui a pour but de générer une liste de taille n. Cette liste de taille n va être notre chemin, c'est-à-dire qu'elle va être un ensemble de villes. `distance_euclidienne(x1, y1, x2, y2)` permet de calculer la distance entre deux villes. `lire_fichier(fichier)` sert à lire les coordonnées de la ville à partir du fichier donné en argument. `distance_chemin(n, fichier, chemin)` prend une liste, par exemple le chemin, et donne la distance du point $(0, 0)$ en traversant tout le chemin et en retournant vers $(0, 0)$. `generer_listes_voisines(lst)` génère toutes les listes voisines possibles d'une liste lst donnée en argument en utilisant la méthode `2-swap`. `meilleur_voisin(chemin)` donne le meilleur voisin du chemin donné en argu-

ment. Enfin, les deux méthodes `meth_hill_climbing(n, chemin, max_depl)` et `meth_tabou(n, chemin, max_depl)`.

3 Programme ZIMPL

En ce qui concerne le code ZIMPL que j'ai utiliser pour modeliser le probleme de voyageur de commerce , il est compose de plusieurs parties :

3.1 Lecture des données du fichier

Le fichier spécifié dans la variable fichier est lu.
La première ligne du fichier indique le nombre de villes (nb), et les lignes suivantes contiennent les coordonnées des villes.

3.2 Définition des ensembles et des paramètres

L'ensemble I représente l'ensemble des indices des villes.
L'ensemble Villes contient les Ids des villes.
Les paramètres a et b stockent les coordonnées des villes.

3.3 Variables de decision

u est un tableau indicé par les indices des villes, utilisé pour déterminer l'ordre de visite des villes.
x est une variable binaire indiquant si le trajet passe de la ville i à la ville j.

3.4 Fonction de distance

La fonction d(i,j) calcule la distance euclidienne entre les villes i et j.

3.5 Fonction objectif

L'objectif est de minimiser la distance totale parcourue.

3.6 Contraintes

les contraintes pour garantire que une ville est visitée une seule fois et pour eviter les cycles.

4 Résultats

4.1 Fichier tsp50.txt

En testant ce fichier la meilleur solution que j'ai obtenu c'est 687.5210354674887 avec la sequence [16, 17, 18, 8, 47, 46, 5, 7, 3, 1, 11, 13, 15, 12, 50, 20, 24, 26, 22, 49, 19, 23, 25, 10, 14, 48, 9, 6, 4, 2, 44, 40, 39, 35, 30, 32, 42, 43, 45, 41, 37, 36, 38, 28, 27, 29, 31, 33, 34, 21] j'ai remarque que peut etre j'obtient une distance inferieur a ce que j'ai obtenu parce que a chaque fois quand j'augmente `nbdepl` j'obtient une distance inferieur.

4.2 Table recap

nb_points	valeur
Tsp5	194.04
Tsp25	376.75
Tsp50	687.52
Tsp101	2751.73

5 Conclusion

En conclusion, les résultats obtenus dans la résolution du problème du voyageur de commerce ne sont pas nécessairement les optimums globaux en raison de la complexité intrinsèque de ce problème NP-complet. La recherche d'une solution optimale peut s'avérer impraticable pour des instances de taille significative en raison du nombre exponentiel de permutations possibles.