Algorithmique avancée - Projet 2 : Résolution du "Travel Salesman Problem" avec de la recherche avec méta-heuristique et de la PLNE

Dorfner François

23 décembre 2023

1 Recherche avec méta-heuristique

Language utilisé : Python3

1.1 Méthodes et voisinnages implémentés

J'ai implémenté le "steepest hill climbing" (SHC), avec et sans redémarrage. J'ai également implémenté la version de SHC avec une liste de tabou de taille paramétrable. J'ai implémenté le voisinage par swap et le voisinnage "opt". J'ai testé ces méthodes sur tous les fichiers .tsp avec :

- avec les deux types de voisinnages ;
- avec différents nombre de déplacements max ;
- avec différents nombre de redémarrages pour la version avec redémarrage ;
- avec différentes tailles de liste tabou pour la version avec tabou.

2 PLNE

2.1 Modèle mathématique

- Variables :

  Avec N villes et $i, j \in \{1, 2, ...N\}$ pour les indices des villes.

  Coordonnées des villes : $x_i \text{et} y_i$

  Distance de la ville i à la ville j : $d_{i,j}$

  Présence d'un passage de la ville i à la ville j : $z_{i,j}$

  $d_{i,j}$ représentera la distance euclidienne entre les villes i et j, et $z_{i,j}$ sera un booléen à 1 si il y a un passage de la ville i à la ville j.

- Objectif :

$$\text{Minimiser} \quad \text{dist} = \sum_{i,j} d_{i,j} z_{i,j}$$

- Contraintes :

$d_{i,j}$ correspond à la distance entre les villes i et j : $\forall i, d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (1)

Toutes les villes sont visitées une seule fois : $\forall i, \sum_j z_{i,j} = 1$ et $\forall j, \sum_i z_{i,j} = 1$ (2,3)

Il faut noter que cette modélisation peut être optimisée : il y a beaucoup de symétries : une fois une séquence trouvée, elle peut s'effectuer dans un sens ou dans l'autre et être décalée.

3 Résultats

3.1 Méta-heuristique

Avec la méta-heuristique (SHC avec liste de tabou de taille 20), j'ai trouvé comme meilleure solution 401,98. La distance parcourue est de 47 points. Points parcourus par cette solution :

[24, 22, 0, 9, 13, 11, 10, 12, 15, 16, 17, 47, 14, 2, 6, 7, 8, 46, 4, 45, 5, 3, 1, 44, 43, 40, 36, 35, 37, 38, 39, 42, 41, 34, 31, 29, 27, 26, 28, 30, 32, 33, 20, 49, 19, 18, 48, 21, 23, 25].

J'ai fait des essais de SHC avec le voisinnage opt et la liste tabou et après avoir testé plusieurs tailles de liste pour les différents problèmes, j'ai vu que fixer la taille de la liste à un tiers du nombre de villes était souvent suffisant pour trouver la meilleure solution.

	tsp5	tsp25	tsp50	tsp101
Taille liste tabou	1	8	16	33
Temps avec voisinnage opt (s)	0.263	0.405	3.43	51.97
Valeur meilleure solution avec voisinnage opt	194.04	259.70	403.69	707.37
Valeur meilleure solution avec voisinnage swap	194.04	400.91	875.75	1147.69