Problème du voyageur de commerce avec la méthode de la recherche tabou

Chabrol Agathe, Courbot Antoine, Zahiri Oumaima

Mai 2022

1 Introduction

La méthode méta-heuristique qui nous a été assignée dans le cadre du **voyageur de commerce** est celle de la "Recherche Tabou". On a choisi de rester sur la version classique de cette méthode vu qu'elle donnait des résultats satisfaisants et que l'amélioration essayée permettait une convergence vers le mauvais résultat.

2 Formulation du problème

Le but est de trouver le chemin de distance minimale que le voyageur de commerce peut traverser à partir de on point de départ jusqu'à son point d'arrivée (généralement le même point) tout en passant par toutes les villes qu'il souhaite visiter. Il ne doit cependant pas visiter une ville 2 fois (hormis son point de départ).

Le problème du voyageur de commerce peut être modélisé à l'aide d'un graphe constitué d'un ensemble de sommets et d'un ensemble d'arêtes. Chaque sommet représente une ville, et chaque arête symbolise le passage d'une ville à une autre, et on y associe un poids représentant la distance entre chaque ville. Résoudre ce problème reviendrait à trouver un chemin qui passe par chaque ville une fois et d'une longueur minimale.
Dans le cadre de la Recherche Tabou, nous allons avoir en entrée: la configuration initiale ainsi qu'une liste "tabou" vide. Et en sortie nous avons : la meilleure configuration parmi toutes celles explorées.

Modélisation mathématique du problème :

Etant donné un graphe orienté G = (V,R), avec V = villes (sommets) et R = routes (arcs)
On cherche un circuit passant par toutes les villes une seule fois et en ayant un

coût minimum :

$$\min \sum_{(i,j)\in R} c_{ij}x_{ij}, \tag{1}$$

$$\sum_{j:(j,i)\in R} x_{j,i} = 1, \forall i \in R, \tag{2}$$

$$\sum_{j:(i,j)\in R} x_{i,j} = 1, \forall i \in R, \tag{3}$$

$$x_{i,j} = 1, \tag{4}$$

si l'arc (i,j) est dans le circuit optimal

$$x_{i,j} = 0 \tag{5}$$

sinon.

(1) Dans chaque ville, le voyageur rentre dans i.
(2) Dans chaque ville, le voyageur sort de i.

3 Algorithme utilisé

A partir d'une configuration initiale choisie, l'algorithme explore différentes solutions grâce à un opérateur de voisinage. La sélection d'une solution par rapport à la configuration initiale doit être la meilleure parmi son voisinage.
Chaque solution choisie est stockée dans une liste tabou. À chaque choix d'une solution parmi un voisinage, la solution choisie est stockée dans une liste tabou.

Le temps que la solution reste dans la liste dépend de la longueur maximale de cette liste, que l'on aura fixée au préalable. À chaque sélection, la nouvelle solution choisie ne doit pas faire partie de cette liste tabou.

Le critère d'arrêt choisi va permettre de déterminer le nombre d'itération qu'il est nécessaire de faire pour pouvoir obtenir notre solution finale.

4 Etude paramétrique

Choix des paramètres :

- Durée de tabou
La durée de tabou correspond à la valeur que l'on assigne à un mouvement interdit dans la liste tabou, elle est à déterminer empiriquement. Si cette durée est trop petite, cela peut conduire à un cycle et on peut ne jamais obtenir de solution optimale (on peut rester bloqué sur un extremum local), dans ce cas, le problème mène donc à une convergence prématurée. Au contraire, si elle est trop grande, elle peut interdire des transformations intéressantes.

Ce paramètre dépend du nombre d'itérations que l'on souhaite effectuer et du nombre de villes que l'on choisit dans les conditions du problème.

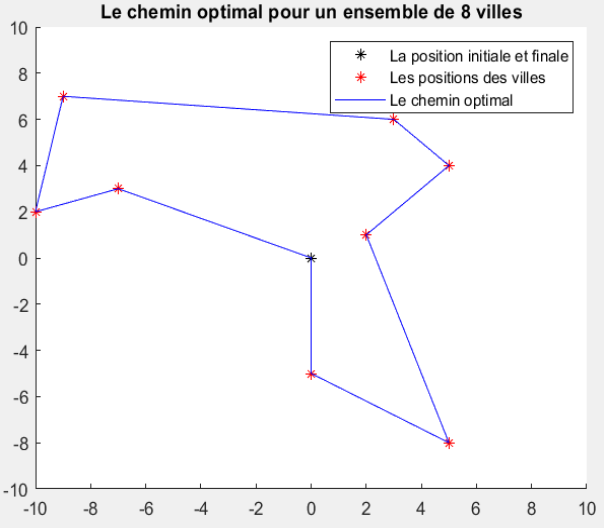
- Le critère d'arrêt

Le critère d'arrêt consiste au nombre d'itérations choisi. Il faut choisir un nombre d'itérations suffisemment grand pour que la solution converge mais pas trop grand pour que ce soit supportable par le CPU et que le temps d'exécution ne soit pas énorme.

5 Exemples

Afin de tester notre algorithme, nous allons illustrer 4 exemples où l'on génère, pour chaque exemple, un certain nombre de villes (en suivant la méthode proposée pendant la séance du CM). On va commencer par des nombres petits afin d'observer le comportement de l'algorithme pour différentes tailles du problème.

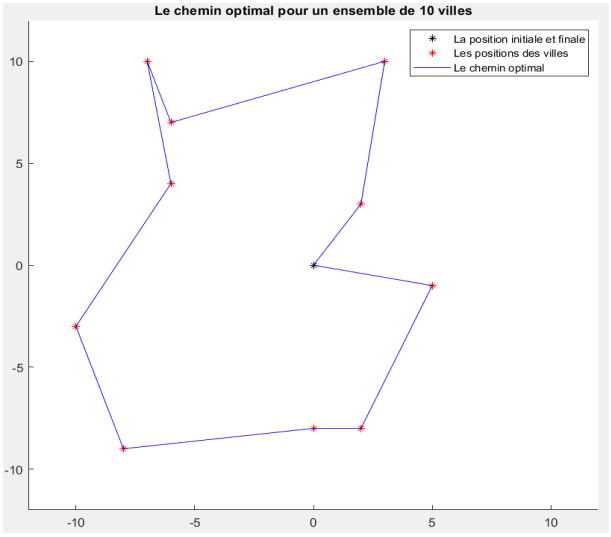
- Exemple d'un chemin à 8 villes :



On génère 8 villes aléatoirement. On a effectué un nombre de 10 itérations avec une durée de tabou de 15. Le chemin optimal de ce problème (le chemin en bleu) est de distance : 55.3075.

Visuellement, on peut dire que le chemin calculé par le programme est optimal.

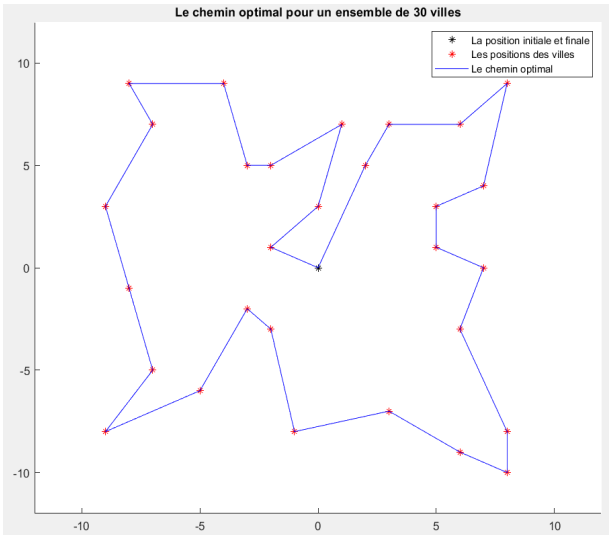
- Exemple d'un chemin à 10 villes :



De même, on génère 10 villes de positions aléatoires. On a effectué un nombre de 10 itérations avec une durée de tabou de 11. Le meilleur chemin obtenu a comme distance : 64.0028.

Comme précédemment, le résultat obtenu est optimal.

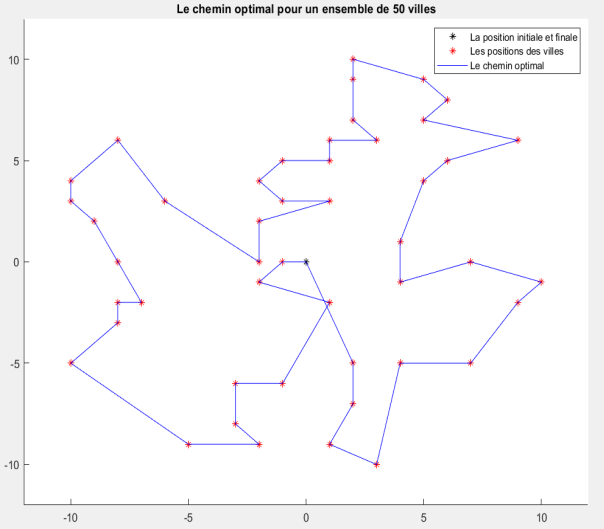
- Exemple d'un chemin à 30 villes :



Pour 30 villes aléatoirement positionnées, on a effectué un nombre de 1500 itérations avec une durée de tabou de 33. Le meilleur chemin obtenu est de distance : 101.4667.

Pour cet exemple, le chemin généré est assez satisfaisant.

- Exemple d'un chemin à 50 villes :



De la même manière, on génère 50 villes positionnées aléatoirement. On a effectué un nombre de 2000 itérations avec un tabuTime = 75. Le meilleur chemin obtenu est de distance de: 118.5729.

Dans cet exemple, le graphe contient deux intersections du chemin, on peut facilement dire que le chemin n'est pas tout à fait optimal et que l'on peut trouver une meilleure solution.

6 Conclusion

Après avoir optimisé les paramètres du problème pour chacun des cas, la recherche tabou donne le chemin optimal au problème du voyageur de commerce pour un nombre petit de villes ($\leq$30). Cependant, l'optimisation des paramètres pour un problème de taille grande reste assez subtile vu que l'on ne possède pas une relation exacte entre la durée de tabou, le nombre d'itérations et la taille du problème; il faut donc tester à la main les résultats des différents paramètres choisis.

Il s'agit de trouver un bon compromis entre les paramètres, chose qui n'est pas évidente à tester à la main avec de nombreuses villes.

Ceci étant dit, les paramètres de cette méthode restent assez peu nombreux comparés à ceux d'autres métaheuristiques (telles l'algorithme de colonie de fourmis et les algorithmes génétiques). De même, l'implémentation de la recherche tabou est plus simple que celle de quelques autres métaheuristiques telles les algorithmes génétiques.

En outre, la recherche tabou est utile pour aider la recherche à s'éloigner des portions de l'espace de recherche déjà visitées et à effectuer une exploration plus étendue.