DENIZOT Hugo F2 20/12/2022

Compte Rendu TP3

CLERMONT AUVERGNE INP ISIMA

Sommaire :

1.Explication du sujet :

Dans ce compte rendu nous allons mettre en place des algorithmes d'optimisation de tournées. Le but est de simulé par exemple une tournée avec différents clients et un camion devant récupérer des quantités chez chaque client. On va donc chercher à savoir quel chemin sera le plus rapide, au vu de la distance des clients ainsi que de la capacité et du nombre de camions disponible, pour réaliser le tour de tous les clients le plus rapidement possible et en évitant les distances. Pour cela nous auront donc ce qu'on appelle des tours géants, ce sera l'ordre dans lequel nous iront chez chaque client durant la journée. Celui-ci sera ensuite découpées en tournée. Chaque tournée finira par le dépôt afin de s'arrêter lorsque le camion sera plein, afin de ne pas faire de voyages chez un client pour finalement ne pas pouvoir tout charger.

2. Présentation des fonctions nécessaires à l'optimisation

a) Génération d'un tour géant :

Cette génération va se faire selon différents modèles que nous expliquerons plus tard lors de la description des algorithmes. Il y aura donc 3 méthodes différentes afin de générer ce tour géant, qui auront chacune un cout différent, et une sera certainement meilleur qu'un autre pour une ville, mais une autre meilleur pour une autre ville.

b) Evaluation d'un tour géant et découpage en tournée :

Cette partie (qui sera appelé SPLIT dans le code) est la fonction principale de tout le sujet. Elle permettra pour un tour géant donné de diviser ce tour géant en différentes tournées et par la suite, évaluer le coût total de cette tournée après différents passages au dépôt pour décharger le camion. Ce sera à la fin de cette fonction que nous auront donc le cout d'un tour géant, qui nous permettra de savoir si c'est dans cet ordre que nous parcourrons nos clients ou non.

c) Conception d'une recherche locale efficace :

La recherche locale elle, va permettre de savoir qu'elle tour géant est le meilleur pour un nombre d'itérations donnée. Elle va réaliser successivement différents tour géant, les splités et les évaluer un par un pour voir quel aura été le meilleur tour géant avec quelle heuristique. Cette fonction est une fonction très importante, cependant, je n'ai pas réussi à la coder. Cette fonction va utiliser, afin de trouver plus rapidement une meilleure solutoins, trois sous fonctions nommées 2OPT pour une tournée, 2OPT pour inter tournée, et le déplacement d'un sommet, qui va permettre de regarder le coût d'un tour géant après n'avoir modifié qu'une liaison dans le tour, une petite partie etc.

d) Conception d'un GRASP() :

Pour la conception d'un GRASP on va utiliser ce qu'on appelle des voisins. On va partir d'un tour géant et d'une solution de base, et à partir de cette solution on va essayer de calculer le cout minimal en plusieurs itérations. On va tirer différents tours géants et pour chaque tour géant on va tirer des tours géants voisins (ce sont des tours géants où seulement 2 clients changent). Après avoir tirés un nombre de voisins déterminés dès le début de la fonction, on va calculer pour chaque tour géant le cout du chemin qu'il propose et voir lequel est le meilleur. Plus le nombre d'itérations de cette fonction est grand, plus le cout minimal retourné devrait converger vers le vrai cout minimal. Pour gagner du temps et éviter de recalculer des tours géants déjà calculés on va passer par une fonction de hachage qui permettra de savoir si le tour géant a déjà été évalué ou non mais cela sera expliqué dans la suite du compte rendu.

3.La description des algorithmes utilisés :

a) Création des tours géants :

Tout d'abord il y a 3 types d'instanciation de tour géants :

-Le premier est le plus simple, celui des plus proches voisins. On va partir du dépôt et parcourir la liste de tous les clients afin de trouver celui qui sera le plus proche. Une fois trouver une répète l'action à partir de lui. Cela va permettre de créer un chemin assez logique cependant on peut vite se retrouver avec un retour vers la fin qui est très éloigné du dépôt, ce qui ne sera pas pratique au moment de la dernière tournée.

-Le deuxième, le tour géant par génération aléatoire. On va partir du dépôt et choisir un nombre défini de points les plus proches (ici 4), et dans ces 4 plus proches, on va tirés aléatoirement un des 4 afin de savoir quelle sera le prochain point. Afin que ça ne soit pas totalement n'importe quoi, on a décidé de faire en sorte que l'algorithme est plus de chances de tirés le 1er, un peu moins le 2ème, un peu moins le 3ème et un peu moins le 4ème.

-Enfin le 3ème tour géant par la méthode des plus proches voisins mais en s'éloignant du dépôt. Cette méthode vue en cours consiste à choisir les 4 points les plus proches du dépôt, et de choisir le point qui s'éloigne le plus du dépôt tant que l'on n'a pas parcouru la moitié des clients. On répète cette méthode à partir du nouveau point et ainsi de suite. Une fois la moitié des clients parcouru, on va chercher dans les 4 plus proches points celui qui nous rapprochera le plus du dépôt.

Ce sont donc les 3 méthodes vu en cours que nous avons implémentés. Il en existe d'autres, qui ont tous leur qualités et leur défaut et qui ont toutes un impact différentes selon le contexte où nous l'utilisons (nombre de clients, distance entres, …).

b) Procédure évaluation d'un tour géant/ SPLIT :

Pour l'évalution d'un tour géant, nous allons parcourir le tour géant de sorte à le parcourir le plus vite possible et avec le moins de tournées possibles. Pour cela nous allons passés par des tournées, qui seront défini par la capacité max d'un camion. L'algorithme va donc parcourir le tour géant en partant d'un premier sommet et en essayant de parcourir un maximum de clients, va définir leur père, tout en faisant attention à ne pas dépasser la capacité maximal du camion. Une fois la liste des clients parcourures, l'algorithme va comparer les pères de chacun afin de voir qui appartient au mêmes clients, et va donc commencer à créer les listes de chaque tournées. Une fois ça de fait, l'algorithme à donc créer les différentes tournées avec pour chacune les clients par lesquels elles passent. Pour finir et pour calculer le cout total, l'algo va parcourir chaque tournée afin de calculer leur cout individuelle, et enfin, va aditionner le cout de chaque tournée afin d'avoir le cout total.

c) Procédure Recherche Locale :

(Pas implémenter dans le code)

d) Procédure GRASP() :

(Pas implémenter dans le code)

e)Proécdure 2OPT tour géant :

Cette procédure va permettre de modifier un tour géant en inversant 2 points, et donc inverser l'ordre dans lequel nous allons parcourir le tour géant entre ces 2 points. Voici un exemple :

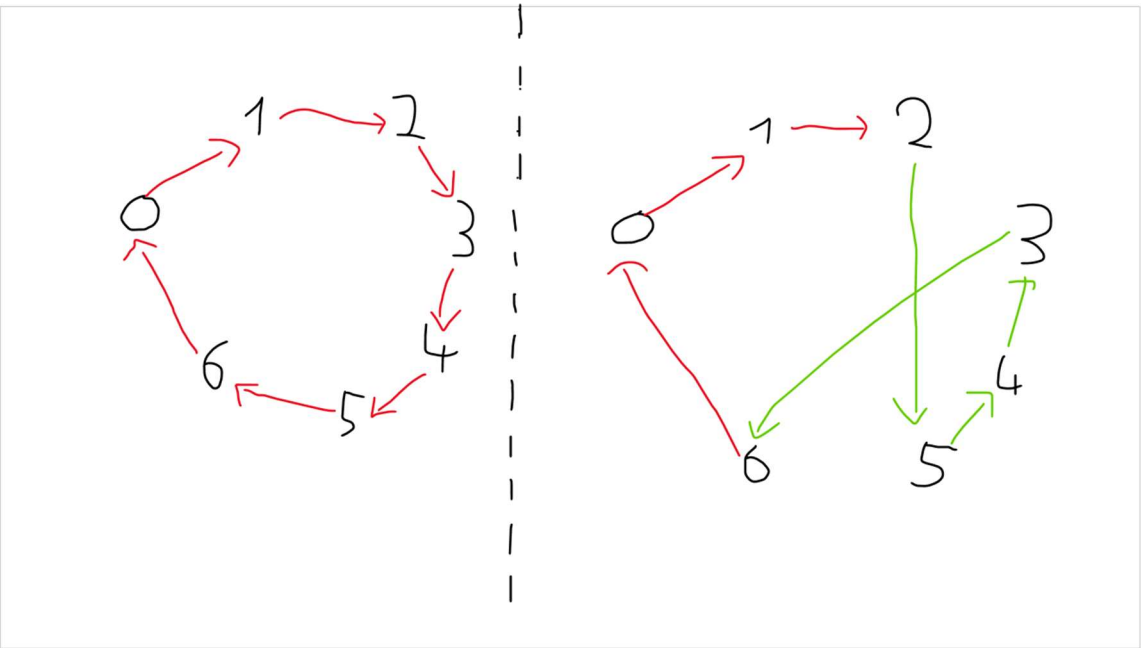

Figure 1 : Explication 2OPT tour géant.

Ici par exemple, nous avons inversé le client 3 et 5, ainsi nous devons passé par le client 4 entre et donc changer de sens entre les 2.

f)Procédure 2OPT inter tournée :

Cette procédure va permettre de modifier le contenu de 2 tournée en les permuttant entre elles à partir d'un point aléatoire pour chacun d'elle.

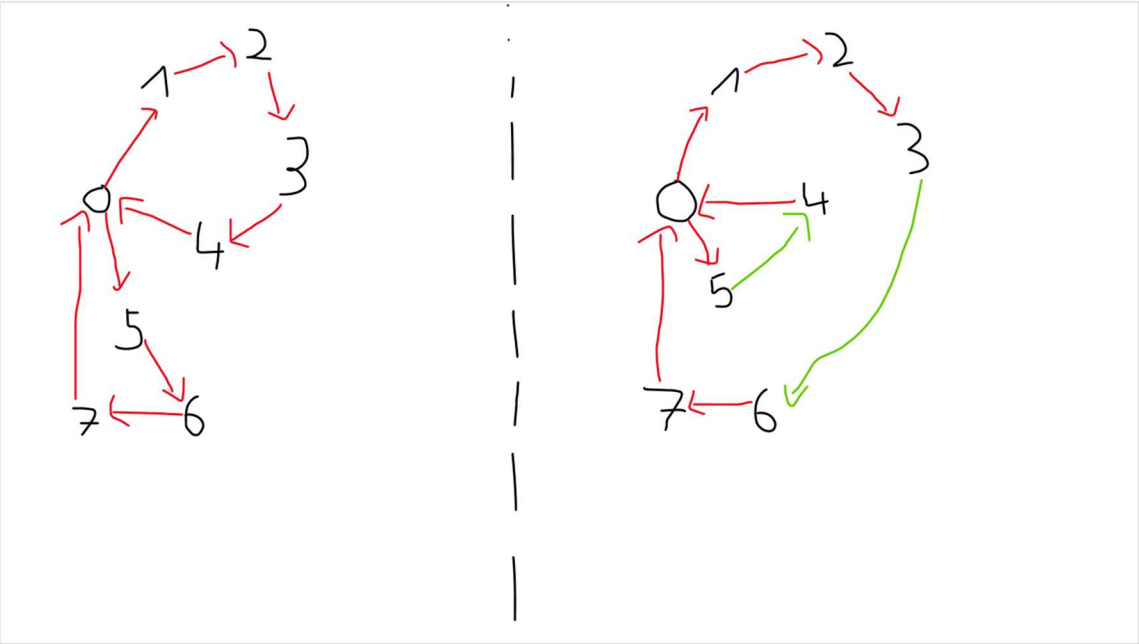

Figure 2 : Explication 2OPT inter-tournée

Ici par exemple, on prendre échanger à partir du 1er point de la deuxieme tournée et du 4ème point de la première, ce qui va nous donner ceci. Le choix des listes et des la positions des sommets que l'on échange se fait aléatoirement dans le code.

g)Procédure déplacement d'un sommet :

Enfin la procédure de déplacement d'un sommet. Celle-ci va se faire aléatoirement comme les autres, nous allons prendre un sommet au hasard dans le tour géant, décaler tout les autres sommets vers la gauche, et le réinsérer à une autre partie du tour géant en décalant le tout par la droite. Cela va donc permettre peut être en changeant seulement un sommet de changer le coût et donc de tomber sur quelque chose de plus optimal.

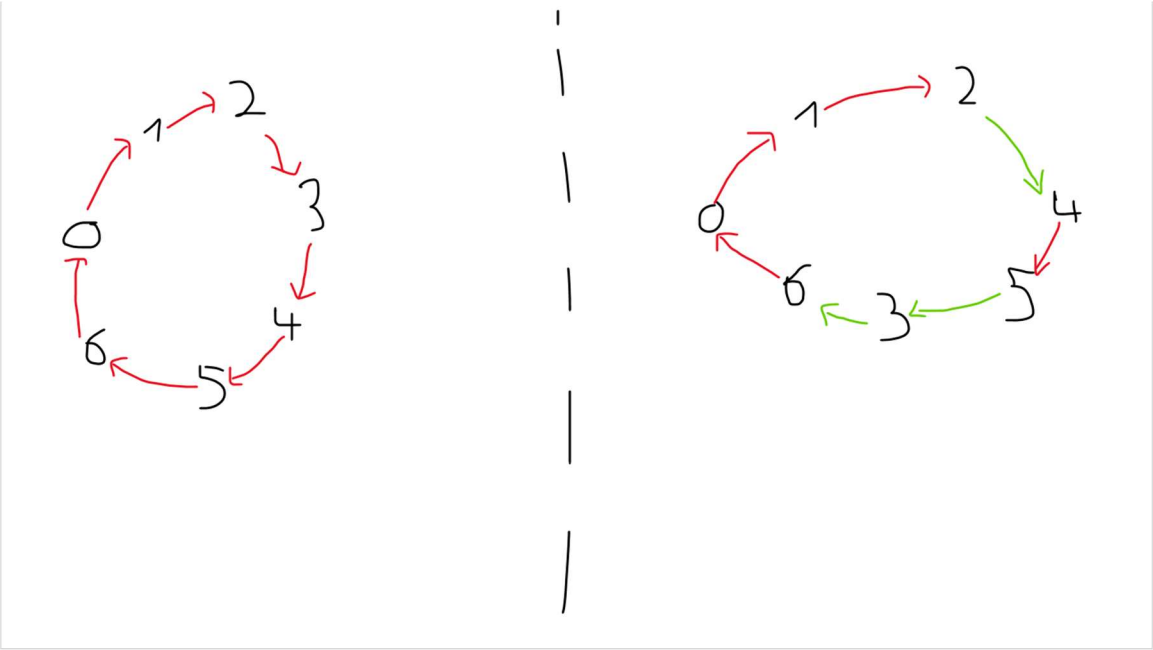

Figure 3 : Déplacement d'un sommet

Ici par exemple, nous avons déplacé le client 3 entre le client 5 et 6.

4.Etude sur une ville :

Etant donné que je n'ai pas de fonction recherche Locale ainsi que de fonction GRASP je ne peux pas tester mon code sur un dizaine de tour géants différents avec quelques modifications, mais pour vous donner une idée du résultat avec les fonctions présentées précédemment, voici le résultat pour le département de Paris avec les 3 heuristiques différentes, en déplaçant un sommet et en faisant un 2 OPT sur le tour géant.

```
Le cout de la solution est 80169

Le cout de la solution apres 2OPT tour geant est 78762
Le cout de la solution apres le deplacement de sommet est 114775
```

Figure 4 : Cout avec heuristique plus proche voisin

```
Le cout de la solution est 75403

Le cout de la solution apres 2OPT tour geant est 78236
Le cout de la solution apres le deplacement de sommet est 77310
```

Figure 5 : Cout avec Heuristique aléatoire

```
Le cout de la solution est 83086

Le cout de la solution apres 2OPT tour geant est 102690
Le cout de la solution apres le deplacement de sommet est 103149
```

Figure 6 : Cout avec plus proche voisin en s'éloignant puis en se rapprochant

On peut donc voir que certaines fois, le déplacement d'un sommet peut changer totalement le cout d'une solution ou alors l'améliorer légèrement, idem avec l'inversion de 2 points dans le tour géant. Il faudrait donc, manipuler ces fonctions plusieurs fois dans une recherche locale ou un GRASP afin d'avoir le cout le plus optimal pour la tournée.