Problème de tournées de véhicules

June 3, 2021

1 Présentation générale du projet

1.1 Thème du projet

Ce projet s'intéresse au problème de Tournées de véhicules (Vehicle Routing Problem). Un certain nombre de véhicules partent d'un dépôt, doivent visiter un ensemble de points de livraison donnés puis retourner à leur dépôt. L'objectif du projet est d'une part de modéliser ce problème, d'autre part de proposer des solutions à ce problème et d'être en mesure d'en évaluer les résultats. Le problème est soumis à un certain nombre de contraintes techniques:

- Les véhicules ont une capacité limitée et ne sont donc pas en mesure de visiter tous les points de livraison
- Une utilité sera affectée à chacun des points de livraison, qui pourra être prise en compte dans le choix des points à livrer en cas de capacité limitée
- Pour les instances les plus difficiles, on associe aux dépôts des fenêtres de temps durant lesquelles doivent avoir lieu les livraisons.

Une solution acceptable consiste d'une part en fournir une affectation des points de livraison aux véhicules en tenant compte des capacités de ces derniers, d'autre part il s'agit d'optimiser les routes de chacun de ces véhicules. Il est laissé au choix des groupes de décider des critères pertinents à optimiser pour le problème. En pratique les objectifs peuvent énormément dépendre du cas d'application, il est donc intéressant de se demander comment des critères d'affectation différents vont affecter les solutions obtenues. A titre d'exemple, on peut chercher à minimiser la distance totale parcourue, qui, dans le cas du problème sans capacité revient toujours à n'utiliser qu'un seul véhicule. Où bien, tenant compte des différentes capacités, on peut chercher à trouver une solution utilisant le petit nombre de véhicules, quitte à avoir des affectations déséquilibrées, ou bien à répartir les utilités entre véhicules le plus équitablement possible. Il n'est pas garanti que la somme des capacités disponibles soit suffisante pour satisfaire l'intégralité des points de livraison, il faudra donc dans cette situation choisir les points à livrer ou non.

1.2 Organisation/évaluation du projet

Le projet pourra être fait au choix en Python ou en Java et durera pendant 5 séances de 3 heures. Le code source devra être envoyé ainsi qu'un rapport. Le projet se fera soit en petit groupe de 3/4 personnes, soit en trois grands groupes. Un ensemble de fichiers tests sera envoyé dans un *.zip*, ces instances seront de difficultés croissantes. Dans le dernier jeu d'instance des fenêtres de temps seront considérées.

Il est attendu une reflexion et une argumentation sur les choix des implémentations, ainsi que des comparaisons qualitatives et quantitatives basées sur l'utilisation de diverses méthodes. Chaque choix de critère d'optimisation sélectionné devra être justifié. L'évaluation principale portera sur la capacité d'appréhender et d'analyser un problème de recherche, puis de trouver et d'implémenter des approches spécifiques pour le résoudre. Enfin il s'agit d'être capable de justifier et d'illustrer (*par des graphes, des tableaux...*) la pertinence de ses résultats.

Le rapport fourni devra expliciter les méthodes et les algorithmes sélectionnés pour résoudre le problème, puis fournir des résultats analytiques et en discuter les performances et leurs limites (par exemple les tailles maximales d'instance pouvant être résolues dans des temps raisonnables...).

Il s'agit donc de:

- **(1):** Comprendre le problème
- **(2):** Lire les données au bon format/les écrire au format demandé
- **(2):** Proposer une (ou plusieurs) heuristiques de construction pour le problème, prenant en compte les différents critères à optimiser ainsi que les contraites. Certaines des instances n'ont pas de contrainte de capacité, donc il ne s'agit dans ce cas là que de minimiser la longueur totale parcourue. Dans le cas où il y a de la capacité (avec un ou plusieurs véhicules), il s'agit de proposer des affectations qui d'une part ont du sens, d'autre part respectent les capacités de chaque véhicule.
- **(3):** Proposer une (ou plusieurs) méthodes de Recherche Locale améliorant la solution constuite initialement (*échange de points entre les véhicules, échange d'arêtes, de sommets...*). Justifier les indicateurs d'améliorant d'une solution.
- **(4):** Evaluer/justifier/juger les résultats obtenus. Pour cela définir des indicateurs (*fraction de points affectés, longueur moyenne parcourue...*). Potentiellement utiliser le solveur linéaire pour obtenir des bornes inférieures/supérieures de certains aspects du problème (*plus petite longueur possible, borne supérieure sur l'utilité pouvant être affectée...*).

1.3 Contenu détaillé du projet

Les premières séances seront consacrées à la compréhension du problème et à l'implémentation de la structure de données. La première séance se concentrera

sur le problème à un seul véhicule sans capacité et donc sur l'optimisation de parcours (*Problème du Voyageur de Commerce*) puis au problème avec plusieurs véhicules à capacités limitées. Le problème à un seul véhicule sans capacité fait parti des problèmes de Recherche Opérationnelle les plus étudiés. Des premières heuristiques seront implémentées et testées. Durant les séances suivantes, des méthodes de Recherche Opérationnelle seront présentées qui pourront être utilisées dans le projet, telles que l'utilisation de certaines méthodes de Recherche Locale. De plus, une introduction à un solveur d'optimisation linéaire sera effectuée, dont l'utilisation pourra être utilisée et valorisée dans le projet.

2 Structure de données

2.1 Notations

Appelons:

- N_V: nombre de véhicules
- N_P: nombre de points de livraisons
- V: la liste de véhicules. On désignera donc le i-ème véhicule par $V[i]$
- P: la liste des points de livraison. On désignera donc le i-ème point de livraison par $P[i]$
- W: pour désigner l'entrepôt d'où doivent partir/arriver les véhicules

Chaque véhicule v est constitué des attributs:

- *index*: indice du véhicule
- *capacity*: capacité du véhicule

Chaque point de livraison p est constitué des attributs:

- *index*: indice du point de livraison
- x: abscisse du point de livraison
- y: ordonnée du point de livraison
- *weigth*: poids affecté au point de livraison
- *utiliy*: valeur d'utilité affectée au point de livraison

On associe à l'entrepôt une abscisse (x) et une ordonnée (y).

2.2 Fichier d'entrée

Le fichier d'entrée sera présenté de la forme suivante, appelons I le numéro de l'instance:

I				
N_V				
N_P				
$W.x$	$W.y$			
$V[0].index$	$V[0].capacity$			
$V[1].index$	$V[1].capacity$			
...	...			
$P[0].index$	$P[0].x$	$P[0].y$	$P[0].weight$	$P[0].utility$
$P[1].index$	$P[1].x$	$P[1].y$	$P[1].weight$	$P[1].utility$
...	...	...	...	...

2.3 Fichier de sortie

Attribuons à chaque véhicule une liste *route* qui contiendra dans l'ordre la liste des points de livraison à parcourir, le paramètre *routeLength* indiquant la longueur de cette route et le paramètre *utility* l'utilité de la route (soit la somme des utilités de chacun des points de livraison). $V[i].route[j]$ désigne donc le j-ème point de livraison du véhicule i. Sur chaque instance qui sera fourni, il sera attendu un fichier de sortie intitulé *output_I.dat* par instance qui sera sous la forme:

I		
0	$V[0].routeLength$	$V[0].utility$
$V[0].route[0].index$		
$V[0].route[1].index$		
$V[0].route[2].index$		
...		
1	$V[1].routeLength$	$V[1].utility$
$V[1].route[0].index$		
$V[1].route[1].index$		
...		