Evolution du VRP

Rapport SAE 6.01

I. Formulation mathématique

A. Modélisation Mathématique

1. Rappels des données

Le problème abordé dans ce sujet est basé sur le problème de tournées de véhicules (plus connu sous le nom de “Vehicle Routing Problem” ou “VRP” en anglais) . Pour répondre à ce problème, différentes données sont disponibles, mais il est important de rappeler les données que nous utilisons :

- V, ensemble des véhicules.
- C, ensemble des clients.
- D, dépôt (et son id).
- $Dist_{ij}$, distance entre le nœud i et j.
- x_{ij}^{v}, tableau des chemins pris ou pas.
- $demande_{i}$, demande du client i.
- $Qmax$, capacité maximum des véhicules

2. Fonction objectif : minimiser

$$Z = \sum_{v \epsilon V} \sum_{i \epsilon C} \sum_{j \epsilon C} (Dist_{ij} * x_{ij}^{v})$$

Ici, nous cherchons à minimiser la somme des distances des trajets (i et j) de chaque véhicule (v). Pour cela, il suffit donc de calculer la somme des trajets ($\sum_{v \epsilon V} \sum_{i \epsilon C} \sum_{j \epsilon C} (Dist_{ij} * x_{ij}^{v})$) réalisés pour chacun des véhicules.

3. Contraintes :

Notre solution doit prendre en compte les contraintes ci-dessous :

1. Un véhicule qui quitte le dépôt doit retourner au dépôt à la fin de sa tournée
2. Un véhicule doit visiter un client une et une seule fois
3. Au moins un véhicule est utilisé pour la construction des tournées
4. Un client doit être visité une et une seule fois par un véhicule
5. L’élimination des sous-tours, c'est-à-dire éviter les boucles et obtenir un circuit hamiltonien.

6. La capacité du véhicule ne peut pas être dépassée

B. CPLEX et modélisation mathématique

1. Fonction objective

expliquer comment tout fonctionne (MTZ, recherche de solution ?)

2. Contraintes

1. Un véhicule qui quitte le dépôt doit retourner au dépôt à la fin de sa tournée
   Voici ci-dessous notre modélisation mathématique de la contrainte ainsi que le code CPLEX qui la représente.

$$\sum_{i \epsilon C} x^{v}_{iD} \leq 1$$

```
// Chaque véhicule doit revenir au dépôt s'il sort
forall(v in Vehicules, i in Noeuds : i != idDepot) {
  sum(j in Noeuds : j != i) x[j][i][v] == sum(j in Noeuds : j != i) x[i][j][v];
 }
```

2. Un véhicule doit visiter un client une et une seule fois
3. Un client doit être visité une et une seule fois par un véhicule

```
// Tous les clients doivent être visités une fois
forall(i in Noeuds : i != idDepot)
  sum(v in Vehicules, j in Noeuds : j != i) x[i][j][v] == 1;
```

ces contrainte CPLEX a été rassembler pour

4. Au moins un véhicule est utilisé pour la construction des tournées

```
// Initialisation de MTZ au dépôt
forall(v in Vehicules) {
  u[idDepot][v] == 0;
}
```

5. Elimination des sous tours

```
// Contraintes MTZ pour éviter les sous-tours
forall(v in Vehicules, i in Noeuds : i != idDepot, j in Noeuds : j != idDepot && i != j) {
  u[j][v] >= u[i][v] + Demande[j] - Qmax * (1 - x[i][j][v]);
}
```

6. La capacité du véhicule ne peut pas être dépassée.

```
// Capacité restante doit rester dans les limites [0, Qmax]
forall(v in Vehicules) {
  sum(i in Noeuds, j in Noeuds) Demande[i] * x[i][j][v] <= Qmax;
}
```

3. Exécution

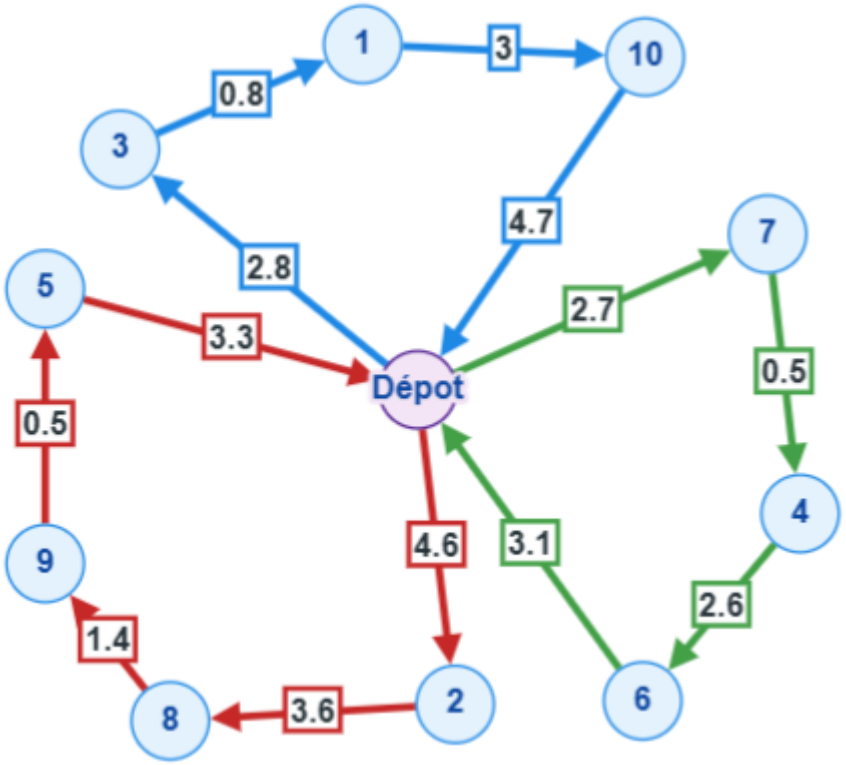
Nous avons utilisé les logiciels et machines de l'iut.
Notre CPLEX met environ moins de 5 secondes avant de se résoudre.
Les fichiers .mod et .dat que nous avons écrits sont disponibles sur Github.

C. Affichage des résultats

1. Livraison des cantines

Nous avons executer notre programme CPLEX avec les données de la livraison des cantines, le résultat obtenu se trouve ci-dessous :

```
// solution (optimal) with objective 31.3
Résultats de l'optimisation:
==============================
 Véhicule 1:
 ------------

Ce véhicule n'a pas été utilisé
==============================
 Véhicule 2:
 ------------

  Tournée      : Dépôt -> 3 -> 1 -> 10 -> Dépôt
  Qté déposées : 100 -> 89 -> 29 -> 3 -> 3
  Distance parcourue : 04.7 -> 3 -> 0.8 -> 2.8
  Capacité utilisée : 97
==============================
 Véhicule 3:
 ------------

  Tournée      : Dépôt -> 2 -> 8 -> 9 -> 5 -> Dépôt
  Qté déposées : 100 -> 56 -> 29 -> 19 -> 1 -> 1
  Distance parcourue : 03.3 -> 0.5 -> 1.4 -> 1.3 -> 4.6
  Capacité utilisée : 99
==============================
 Véhicule 4:
 ------------

  Tournée      : Dépôt -> 7 -> 4 -> 6 -> Dépôt
  Qté déposées : 100 -> 68 -> 53 -> 33 -> 33
  Distance parcourue : 03.1 -> 2.6 -> 0.5 -> 2.7
  Capacité utilisée : 67

==============================
==============================
Statistiques globales :
  Nombre total de véhicules utilisés : 3 / 4
  Distance totale parcourue : 31.3
```



Avant de résoudre des problèmes plus complexes, nous avons cherché des problèmes plus simples pour s'assurer d'avoir compris la logique de la résolution du problème. Nous avons donc commencé par aborder le problème de la livraison des cantines.

Tout d'abord, nous avons essayé de visualiser le graphe grâce à un tableau :

	Z (Dépôt)	A	B	C	D	E	F	G	H	I	J
Z (Dépôt)	0	2.7	4.6	2.8	3.0	3.3	3.1	2.7	5.1	3.9	4.7
A	2.7	0	3.1	0.8	1.8	2.5	4.2	1.4	3.6	2.5	3.0
B	4.6	3.1	0	3.3	4.4	1.7	6.8	4.1	1.3	1.7	1.4
C	2.8	0.8	3.3	0	1.9	2.0	4.0	1.5	3.8	2.8	3.2
D	3.0	1.8	4.4	1.9	0	3.4	2.6	0.5	4.7	4.7	4.1
E	3.3	2.5	1.7	2.0	3.4	0	5.8	3.0	1.8	0.5	2.6

F	3.1	4.2	6.8	4.0	2.6	5.8	0	3.0	7.4	6.1	7.6
G	2.7	1.4	4.1	1.5	0.5	3.0	3.0	0	4.6	3.7	4.3
H	5.1	3.6	1.3	3.8	4.7	1.8	7.4	4.6	0	1.4	2.8
I	3.9	2.5	1.7	2.8	4.7	0.5	6.1	3.7	1.4	0	2.8
J	4.7	3.0	1.4	3.2	4.1	2.6	7.6	4.3	2.8	2.8	0

II. Développement de l'app

A. Explication du scanneur

B. je sais pas