Module MAOA :
Modèles et Application en Ordonnacement et optimisation
combinAtoire

AUTOUR DU PROBLÈME DES TOURNÉES DE TECHNICIENS

Lachiheb Sarah et Cassandre Leroy

Sorbonne Université Pierre et Marie Curie
Master Androide 2018-2019

Table des matières

1 Introduction

Les problèmes d'optimisation combinatoire en recherche opérationnelle que nous allons traiter tout au long de ce rapport sont le problème des tournées de véhicules et des tournées de techniciens. La combinatoire des solutions associées à ces problèmes est tellement importante qu'il est difficile de déterminer de façon exacte la solution optimale dans des temps acceptables. Ces problèmes sont tellement combinatoires que la simple énumération des solutions est impossible dès que la taille des données augmente. Cela pointe le coté NP-complet du problème de tournée, c'est-à-dire qu'il n'existe pas d'algorithme de résolution exacte en un temps "raisonnable", dit polynomial.

Dans ce projet sera développé un logiciel, permettant de résoudre des instances du problème présentes sur le site : `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/` en tenant compte du temps d'exécution et surtout de la qualité de la solution obtenue.

Dans une première partie, nous nous intéresserons au problème de tournées de véhicules (VRP) et dans une deuxième partie au problème de tournée de techniciens étant une extension de la VRP avec la prise en compte des compétences du techniciens. Chaque partie mettra en œuvre trois types de programme : une résolution heuristique, une résolution exacte et une évaluation expérimentale.

2 Le problème de tournée de véhicules (VRP)

2.1 Principe

Le problème VRP (Vehicle Routing Problem) est défini par un graphe non orienté complet G (V, R). L'ensemble des nœuds V comprend un dépôt (nœud 1), où se situe un ensemble de véhicules identiques de capacité Q avec un nombre de véhicules disponibles imposé, et n clients avec des demandes q_i. Chaque arêtes $[i, j]$ de l'ensemble R représente un chemin entre les nœuds i et j dans le graphe. Le coût c_{ij} de chaque arête représente une distance en terme de kilométrage. L'objectif consiste à déterminer un ensemble de tournées permettant de desservir tous les clients, tout en minimisant le nombre de kilomètres parcourus par les véhicules. Chaque tournée commence et se termine au dépôt, par un véhicule dont la charge totale ne dépasse pas Q.

Le VRP est NP-difficile car une tournée doit résoudre un problème du voyageur de commerce (TSP) , connu pour être NP-Difficile. En effet, on doit passer par deux phases pour construire une tournée qui sont : affecter un client à chaque

véhicule et dans chaque tournée, résoudre un TSP pour trouver le chemin le plus cours entre chaque client de la tournée. Les figures 1 et 2 représentent un exemple d'instance "A-n32-k5" (32 sommets et 5 véhicules) du VRP résolue graphiquement.

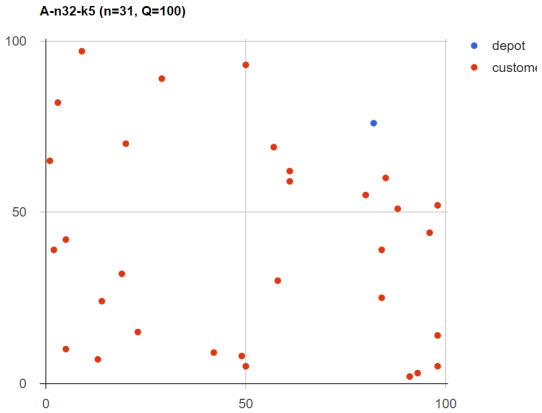

FIGURE 1 – Instance non résolue (CVRPLIB).

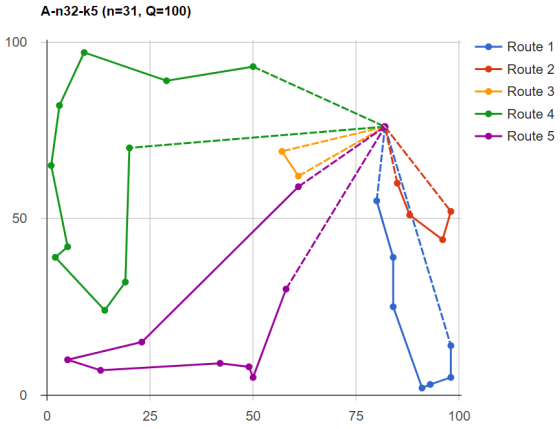

FIGURE 2 – Instance résolue (CVRPLIB).

2.2 Résolution par méthode approchée

La métaheuristique que nous allons aborder dans ce projet, est une métaheuristique de type GRASP/ELS[1]. C'est une hybridation entre deux métaheuristiques :

— GRASP.

— Une recherche locale évolutionnaire de type ELS.

Les métaheuristiques utilisées sont dite à parcours, déterminent une suite de solutions traçant une trajectoire dans l'espace des solutions. Le problème de VRP est extrêmement combinatoire, tellement que toute les métaheuristiques vraiment efficaces incluent des recherches locales.

Dans notre projet, nous allons relaxer la capacité des véhicules pour explorer l'espace de solution du TSP puis appliquer une procédure de découpage pour en déduire des solutions réalisables pour le problème initial.

2.2.1 Méthode GRASP

La méthode GRASP consiste à construire une solution par technique gloutonne correspondant à une phase de construction puis d'amélioration avec une recherche locale tant qu'une condition d'arrêt n'est pas satisfaite. Elle permet d'explorer des

zones très variées de l'espace des solutions en partant de différentes solutions de départ.
La méthode GRASP est composée de deux phases qui sont une phase d'initialisation randomisée et d'une phase d'amélioration.

2.2.1.1 Initialisation d'une heuristique de construction

L'heuristique de construction que nous allons utiliser pour résoudre le problème de tournées de véhicules est le **Route first - Cluster seconde**. Il consiste en la construction d'un tour géant sur l'ensemble des clients à l'aide d'un **algorithme glouton de plus proche voisin**, pour résoudre le problème du voyageur de commerce de manière approchée, puis à découper ce tour en tournées respectant la capacité d'un véhicule avec **l'algorithme Split**.

Route first : Correspond à la construction d'une solution au problème du voyageur de commerce, qui est une méthode de construction gloutonne, basé sur une technique des K plus proches voisins randomisés, dans notre projet c'est 5 voisins qui sont utilisés. Elle se construit par création d'une liste ordonnée de K-client par critère d'éloignement par rapport au dernier client inséré. Puis, on choisit de manière aléatoire un élément de la liste afin de le rajouter dans un vecteur solution. Ce qui nous rend un tour géant contenant tout les sommets client en formant un cycle à partir du sommet dépôt.

Pour le moment, nous n'avons pas une solution du VRP, car il nous faut déterminer les tournées de chaque véhicule. Pour découper, le tour géant en plusieurs tournées, nous allons utiliser l'algorithme de Split.

La figure 3 illustre un graphe complet contenant 5 clients de A à E et un dépôt. Sur ce graphe complet est appliqué l'algorithme glouton des k plus proche voisin avec k 1, illustré par les figures 3 à 9. La figure 10 est le tour géant obtenu par l'algorithme qui est la résolution d'un TSP sur tout les clients.

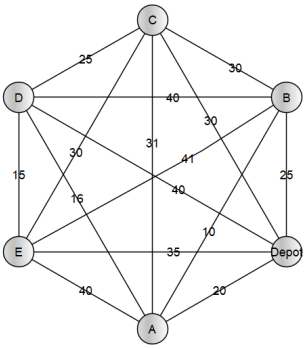

FIGURE 3 – Graphe complet.

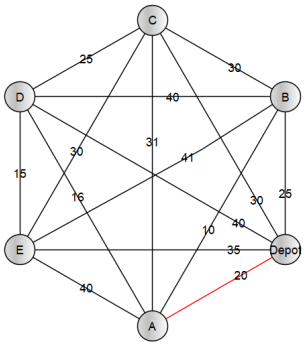

FIGURE 4 – Plus proche voisin du dépôt.

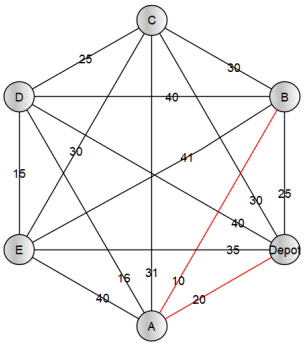

FIGURE 5 – Proche voisin de A.

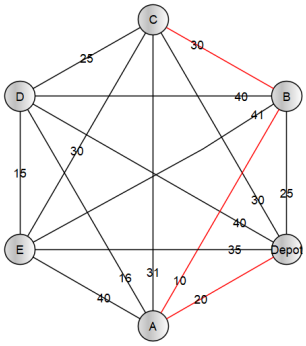

FIGURE 6 – Proche voisin de B.

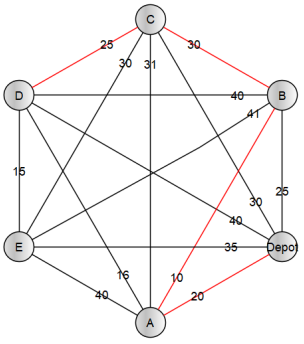

FIGURE 7 – Pproche voisin de C.

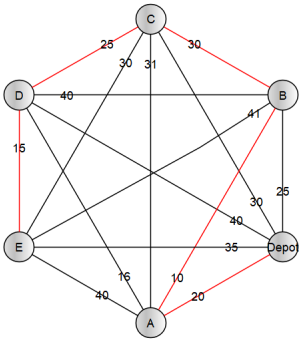

FIGURE 8 – Proche voisin de D.

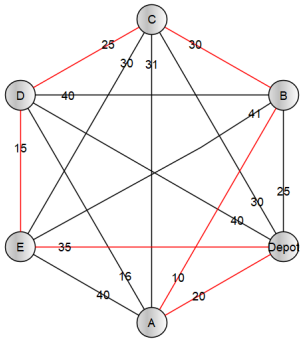

FIGURE 9 – Proche voisin de E.

Cluster seconde : Le tour géant construit, on cherche à trouver les différentes tournées pour construire un élément de l'espace des solutions. Pour atteindre ce but, il faut entreprendre un découpage du tour géant en tournées respectant les capacités des véhicules via l'algorithme de Split.

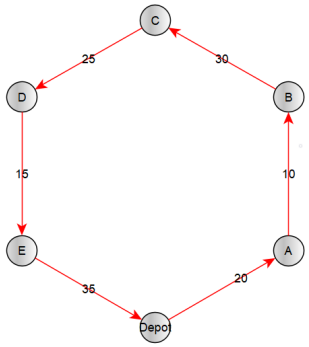

FIGURE 10 – Tour géant.

Le principe de l'algorithme de séparation permet de découper de la meilleur façon une séquence de n clients en tournées de véhicule de capacité limitée.

Pour ce faire, nous allons l'illustrer sous différentes figures. La figure 11 nous montre le tour géant construit précédemment, où chaque client est accompagné d'une étiquette correspondant à leur demande. Chaque véhicule doit prendre soin d'avoir la capacité nécessaire pour contenir l'entière demande du client lors de leurs passages.

Le tour géant considéré dans cette exemple est T $[Dépôt, A, B, C, D, E]$, avec une capacité pour chaque véhicule de $Q = 10$. L'algorithme de séparation à recours a un graphe auxiliaire qui n'est pas généré de manière explicite et qui permet d'identifier toutes les tournées possibles dans l'ordre d'apparition des sommets dans le tour géant, tout en respectant les contraintes de capacité des véhicules illustré par la figure 12.

Dans ce graphe auxiliaire, on peut constater qu'il n'est pas possible d'avoir un graphe avec circuit. Il permet d'utiliser ainsi l'algorithme de Bellman pour calculer le plus court chemin entre les différentes tournées possibles illustré dans la figure 13 et ainsi donner une solution au VRP dans la figure 14.

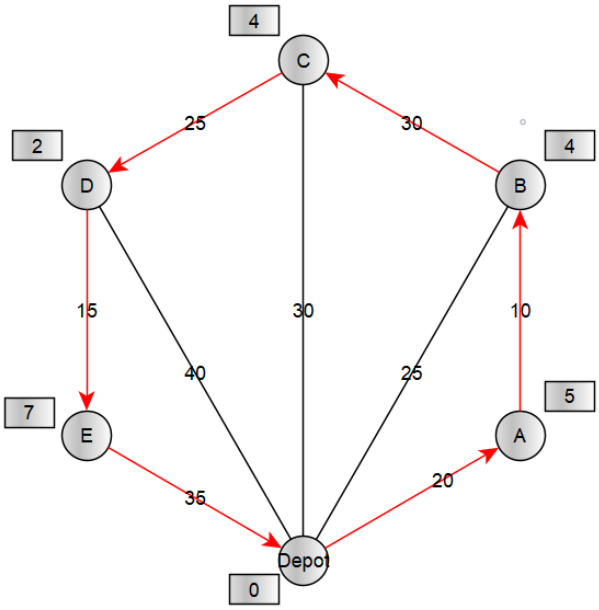

FIGURE 11 – Tour géant sur l'ensemble des clients.

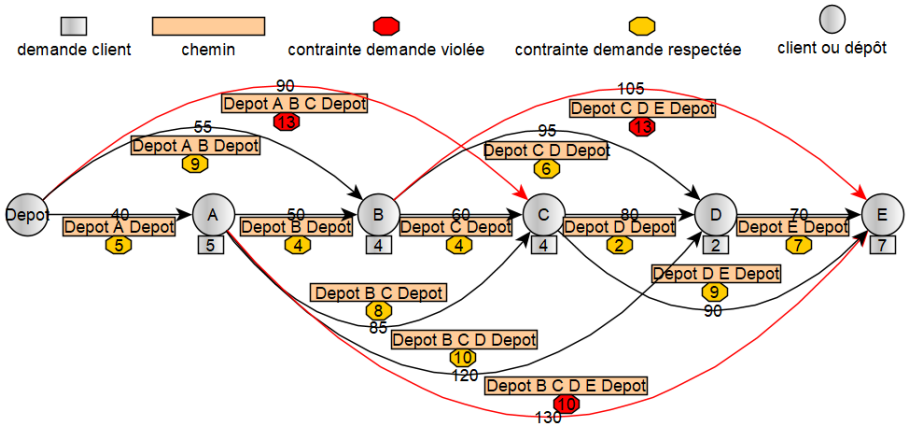

FIGURE 12 – Graphe auxiliaire des tournées possibles.

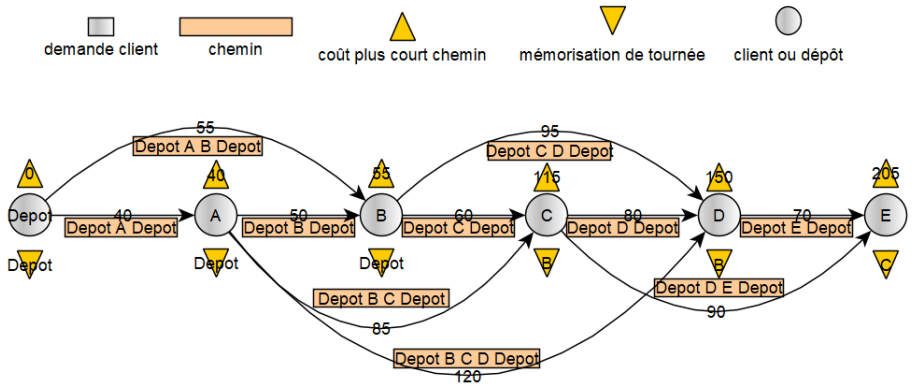

FIGURE 13 – Graphe auxiliaire des tournées possibles.

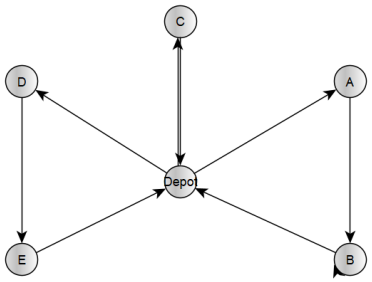

FIGURE 14 – Découpage optimal en tournées.

2.2.1.2 Recherche locale

La phase de recherche locale permet d'améliorer la solution candidate de départ en cherchant la meilleure solution par une succession de transformation. Ainsi, la solution candidate S est remplacé par la meilleure solution $S' \in N(S)$ dans son voisinage. Le processus s'arrête quand il n'est plus possible de trouver une solution améliorante dans le voisinage de S. C'est pourquoi, le choix des transformations appliquées et de leur mise en oeuvre sur la solution est une partie importante de la recherche locale. Si on considère une solution initiale de départ comme celle trouver par l'heuristique de construction précédemment définie et qu'on souhaite l'améliorer à l'aide d'une recherche locale, on obtient à partir de l'espace de voisinage un optimum local en fonction de la transformation utilisée.

Les transformations déployés lors de la recherche d'une solution de qualité pour le problème de tournées de véhicules sont la combinaison du 2-opt-intra-tournée, 2-opt-inter-tournée et du swap-externe-tournée. Ces types de recherche locale sont effectuées sur l'ensemble des tournées identifiées après l'exécution de l'algorithme de Split.

2-opt-intra tournée : le principe de la transformation de 2-opt-intra est d'itérer sur l'ensemble des arcs de la tournée et pour chaque arc sélectionné appelé (u, v) et (x, y) avec v $\not{x}$ et u $\not{y}$, le remplacer par les arcs (u, x) et (v, y) respectivement. Ce qui entraîne l'inversement des sommets compris entre v et x. Il se traduit par la variation de coût $\Delta\ c_{ux}\, c_{vy} - c_{uv} - c_{xy}$, illustré par la figure 15,16 et 17 avec la tournée 1 (T1).

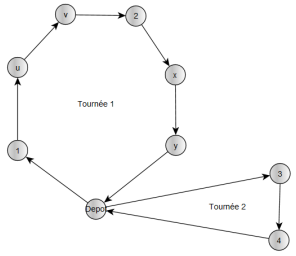

FIGURE 15 – T1 initiale.

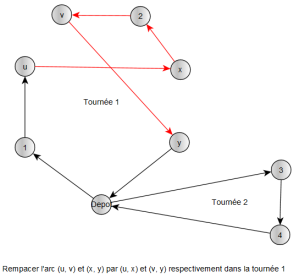

FIGURE 16 – T1 2-opt-intra.

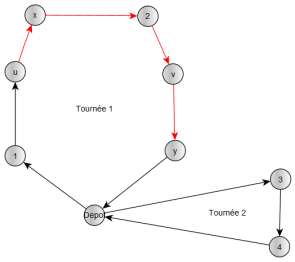

FIGURE 17 – T1 replacement des sommets.

2-opt-inter tournée : la transformation de 2-opt-inter travaille sur deux tournées différentes, comme dans l'exemple sur la tournée 1 (T1) et la tournée 2 (T2) de la figure 18. Pour chaque arc, on prendra ici (u, x) de la tournée 1, on itère sur l'ensemble des arcs de la tournée 2 nommé (v, y). La transformation remplace l'arc (u, x) et (v, y) par (u, y) et (v, x) entraînant une variation de coût de $\Delta\ c_{uy}\, c_{vx} - c_{ux} - c_{vy}$. Le 2-opt-inter tournée est illustré par les figures 19,20 et 21.

Swap-externe tournée : Le Swap externe itère de deux en deux sur des tournées de position successives comme illustré par les figures 22,23 et 24. Elle permet d'échanger deux clients de tournées voisines, afin de trouver le meilleur candidat dans une tournée et position précise en fonction de la configuration des autres tournées. Le Swap externe donne la possibilité de générer des tournées vides donc de réduire le nombre de véhicule utilisés. Le principe de la transformation réside sur l'itération de chaque sommet i de la tournée 1, en incorporant toutes les combinaisons d'insertion possibles d'une position de ce sommet i dans la tournée 2 comme figuré dans la figure 25 à 28. La recherche locale s'exécute de la tournée 1 vers la tournée 2 puis dans le sens inverse de la tournée 2 vers la tournée 1. En effet, cela permet de visité de manière plus minutieuse le voisinage de la solution du VRP pour converger le plus proche possible vers un optimum global.

Prenant trop de temps car il est exponentielle, il est dû être enlevé.

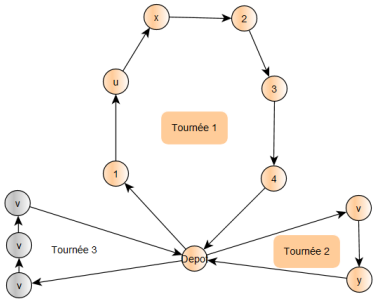

FIGURE 18 – Sélection de T1 et T2.

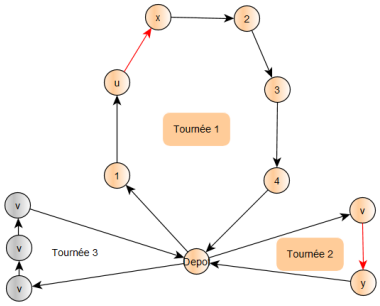

FIGURE 19 – Sélection d'un arc dans T1 et T2.

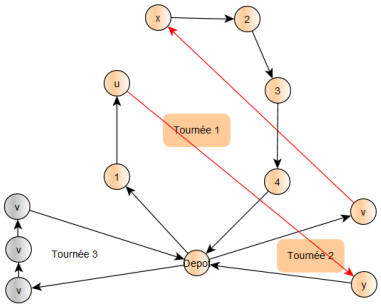

FIGURE 20 – Remplacement de (u,x) par (u, y) et de (v, y) par (v, x).

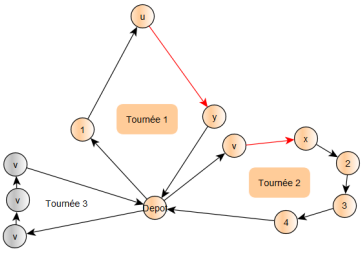

FIGURE 21 – Replacement des sommets.

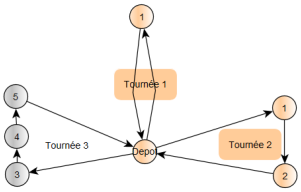

FIGURE 22 – Sélection T1 et T2

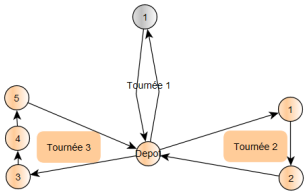

FIGURE 23 – Sélection T2 et T3

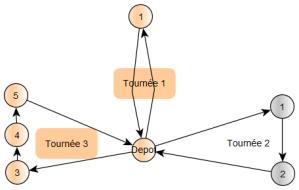

FIGURE 24 – Sélection T3 et T1

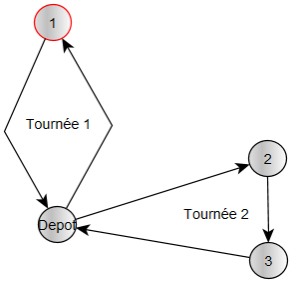

FIGURE 25 – Sélection du sommet i = 1

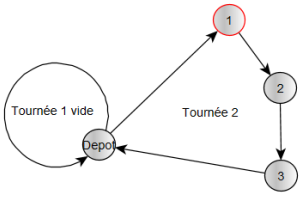

FIGURE 26 – Position i en position 1 dans T2

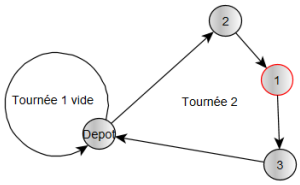

FIGURE 27 – Position i en position 2 dans T2

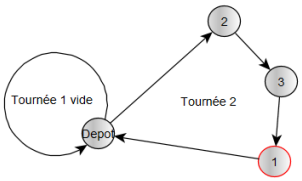

FIGURE 28 – Position i en position 3 dans T2.

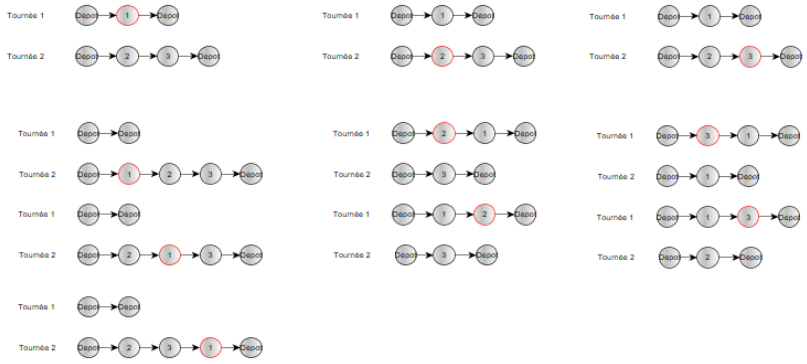

FIGURE 29 – Exemple d'exécution de Swap externe sur deux tournées.

2.2.2 Recherche locale évolutionnaire ELS

La recherche locale évolutionnaire (ELS) a pour vocation d'intensifier la recherche d'une solution de bonne qualité en couvrant les bassins d'attraction adjacents à la solution courante.

En effet, à partir d'une solution candidate du VRP sous la forme d'un ensemble de tournée, l'ELS concatène la suite des tournées pour générer un tour géant. De cette étape, il y a génération de p-solutions enfants sur lesquels on applique une perturbation sous forme de mutation légère (il suffit de sélectionner deux clients dans le tour géant et de les permuter, de là, nous obtenons un fils parmi p) et

nous les obtenons sous forme de tour géant. Le tour géant n'étant pas une solution du VRP, est appliqué alors l'algorithme de Split afin d'obtenir un ensemble de tournée.

Pour acquérir une suite d'optimum locaux, on emploie les trois étapes vues dans la partie 2.2.1.2 et parmi les p enfants, on remplace la solution actuelle par le meilleur enfant, par processus de sélection élitiste, en cas d'amélioration.

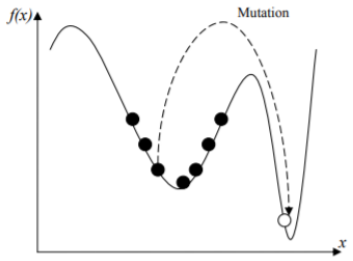

FIGURE 30 – Illustration d'intensification de la solution par mutation

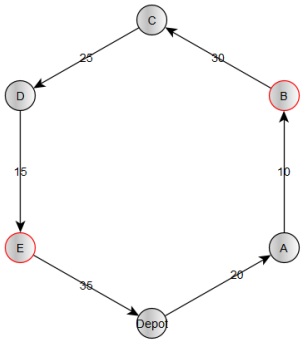

FIGURE 31 – Sélection de deux sommets

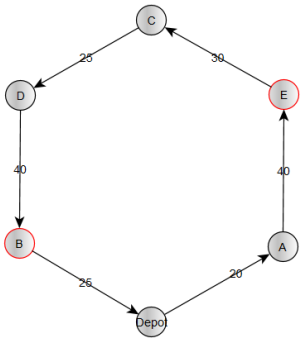

FIGURE 32 – Permutation des clients

2.2.3 Métaheursitique GRASPxELS

La première phase du GRASP repart à chaque itération d'une nouvelle solution de départ construit via une heuristique gloutonne randomisée, permettant ainsi de diversifier l'espace de recherche de la solution. Et la deuxième phase de recherche locale du GRASP est remplacé par la méthode de recherche locale évolutionnaire ELS d'où l'hybridation. Elle permet à la fois de diversifier par mutation plusieurs solutions fils à partir d'une solution courante. Et d'intensifier la recherche en ce limitant plus à un optimum local obtenu par simple recherche locale, mais en explorent un bassin d'attraction autour de cette optimum.

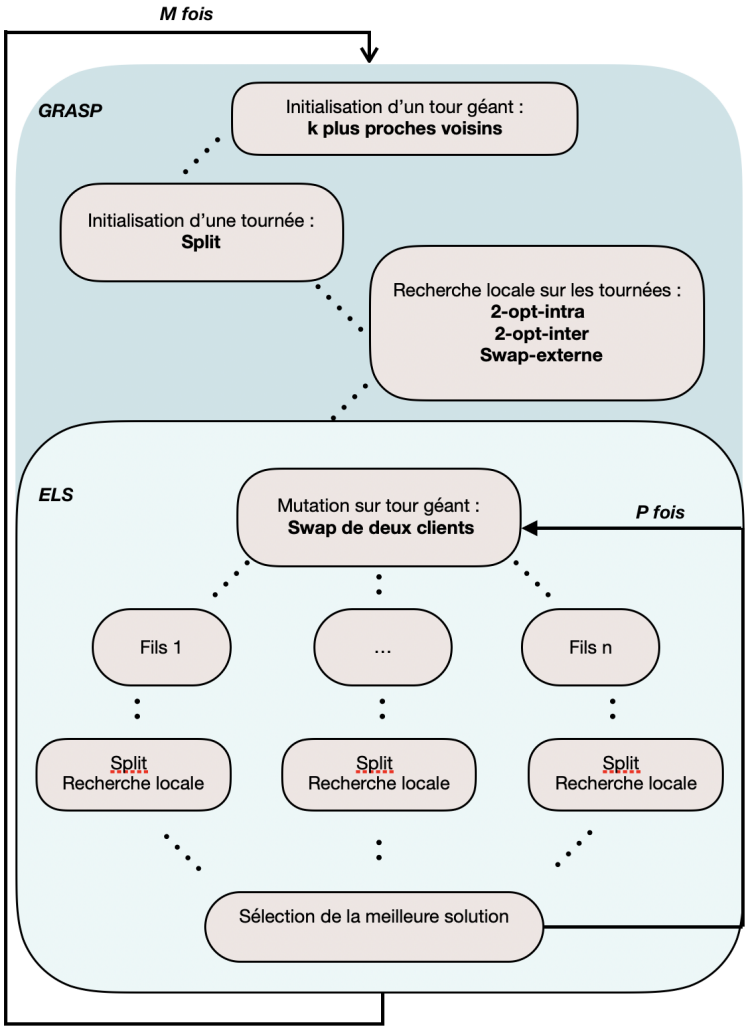

FIGURE 33 – Illustration de la métaheuristique GRASP/ELS

2.2.4 Évaluation expérimentale

L'évolution expérimentale de la méta-heuristique GRASP/ELS possède trois critères à manipuler pour atteindre un espace de solution, de ce le plus optimal possible. Le premier critère est le nombre d'itération de l'heuristique GRASP pour permettre de diversifier l'espace de recherche, le deuxième et troisième critère sont respectivement, le nombre d'itération ELS et le nombre d'enfant produit par itération d'ELS permettent l'intensification de l'exploration du bassin d'attraction. Pour chaque expérimentation, nous effectuons une moyenne du temps d'exécution et de coût pour supprimer le bruit. On extrait par la suite, la solution optimale de l'expérimentation, celle de coût minimal.

Instance	Itération Graps	Itération ELS	Nombre d'enfants	Moyenne du temps (secondes)	Moyenne du coût	Meilleur coût
A-n32-k8	50	50	100	5.350	810.6127	787.082
A-n54-k7	50	30	50	8.347	1253.417	1190.94
A-n80-k10	50	30	50	18.601	1981.104	1894.74
P-n16-k8	50	50	100	1.972	451.940	451.335
P-n22-k2	50	50	100	2.704	222.918	217.852
P-n55-k7	50	50	50	15.233	592.823	575.258
P-n101-k4	50	50	50	104.871	737.132	705.006
M-n101-k10	50	50	50	58.588	888.072	828.277
M-n200-k17	50	50	50	235.345	1554.902	1486.22

Table 1

Dans le tableau 2, nous calculons la pourcentage d'erreur à l'optimal pour la moyenne du coût et pour la moyenne du meilleur coût produit par la méta-heuristique. Il est intéressant de remarquer que la meilleur solution trouver à au plus 7% d'erreurs à l'optimal avec des instances de plus de 100 noeuds. Cela montre l'efficacité de la méta-heuristique à explorer l'ensemble de la l'espace de solution pour converger au maximum vers l'optimum global.

Les méthodes méta-heuristiques sont des algorithmes conçus pour produire une solution réalisable du problème traité, de bonne qualité mais pas forcément optimale. En effet, lorsqu'on regarde les coûts optimaux de chaque instance, en remarque rapidement que l'on atteint jamais l'optimal mais nous n'en sommes pas éloignés, pas plus de 7% d'erreurs sur nos instance. Mais lorsqu'on regarde de plus près la solution trouver (annexe), celle-ci peut être complètement différente des tournées proposées pour cette la solution optimale.
Cela signifie donc que nous pouvons être très proche en coût mais très éloigné en moyen de trouver ce coût (c'est à dire avec des routes très différentes). Cette conclusion, nous amènes à la deuxième méthode de résolution, la résolution exacte.

2.3 Résolution par méthode exacte

Nous allons formuler le problème de tournées de véhicule sous forme PLNE dont le but est de minimiser une fonction linéaire à variables entières en respectant un ensemble de contraintes linéaires. Toutefois, la taille des instances, le nombre de client et de véhicule augmente la durée de calcul en énumération des contraintes et fait augmenter exponentiellement le temps d'exécution. Ainsi, pour résoudre

Instance	Meilleur coût	Pourcentage à l'optimal du meilleur	Pourcentage à l'optimal de la moyenne
A-n32-k8	787.082	0,40	3,2
A-n54-k7	1190.94	1,97	6,0
A-n80-k10	1894.74	6,9	11
P-n16-k8	451.335	0,42	0,29
P-n22-k2	217.852	0,82	3
P-n55-k7	575.258	1,21	4
P-n101-k4	705.006	3.5	8,24
M-n101-k10	828.277	0,97	7.66
M-n200-k17	1486.22	16.6	21.9

TABLE 2

un PLNE en un temps raisonnable est nécessaire d'appliquer des algorithmes de séparation et méthodes de coupes.

2.3.1 Programme linéaire à deux indices orienté avec MTZ

Le premier PLNE étudié est celui à 2 indices orienté avec la formulation compacte basée sur les inégalités de Miller-Tucker-Zemlin (MTZ). Mais, cette formulation ne peut que résoudre des instances de très petite taille. Il s'est alors imposé d'ajouter la formulation de renforcement de la MTZ, qui empêchent les solutions d'avoir des sous-tours. Néanmoins, Le nombre de contraintes d'élimination de sous-tours est exponentiel. Donc, non implémentable sous cette forme.

Pour pouvoir résoudre une telle formulation, on utilise des contraintes linéaires comme des plans de coupes. on cherche une contrainte d'élimination de sous-tours violée par la solution, on l'ajoute au programme linéaire et on résout le nouveau problème. On peut ainsi réinjecter une à une, des contraintes d'élimination de sous-tours. Le principe consiste à ajouter progressivement les inégalités dans la formulation.

Dans le cas, ou la solution est entière ou fractionnaire Cplex va faire, respectivement, appel à ILOLAZYCALBACK ou ILOUSERCUTCALBACK. Cette fonction fait appelle à son tour à un algorithme de coupe. En effet, l'algorithme construit un graphe sous Lemon avec les valeurs de la variable de décision x_{ij}. Puis on applique à ce graphe la coupe minimale [2] de Nagamochi-Ibaraki. Si la coupe comporte plus de 2 clients, alors on a donc identifier une contrainte de sous-tours violée alors on ajoute la contrainte suivante :

$$\sum_{i \in S} \sum_{jS} x_{ij} \geq 1 \ |S| \geq 2$$

Sans compréhension de ma parrt, le PLNE ne fonctionne pas. Cplex appelle sans cesse la fonction ILOUSERCUTCALBACK, sans amélioration de la solution. Donc, nous avons décidé de coder un autre PLNE mais cette fois-ci sans la formulation MTZ.

2.3.2 Programme linéaire à deux indices orienté avec formulation par les coupes

Pour le deuxième PLNE, nous avons choisi d'implémenter la version à deux indices orientés avec la formulation par les coupes. Pour cela, nous avons crée un ILOLAZYCALBACK où l'algorithme détecte chaque cycle [3] ne comportant pas de le noeud dépôt. Si on trouve un cycle sans dépôt alors une contrainte de sous tour est violée, on ajoute la contrainte suivante :

$$\sum_{i \in W} \sum_{j \in \{0,1,\ldots,n\} \setminus W} x_{ij} \geq \lceil \frac{\sum_{i \in W} d_i}{Q} \rceil$$

Mais, ça ne suffit pas. Il faut vérifier si les cycles comportant le dépôt respecte bien la contrainte de capacité. Si ce n'est pas le cas, il y a violation de la contrainte de capacité. On fait alors la somme de chaque arcs présent dans le cycle avec dépôt ne respectant pas la capacité et on ajoute la contrainte de sorte que la somme des arcs soit supérieur à $\lceil \frac{\sum_{i \in W} d_i}{Q} \rceil$.

Ici aussi, malgré de longues heures à faire différentes tentatives je retrouve la même situation que précédemment. A court d'idée pour résoudre notre problème, nous décidons de changer complètement le PLNE dans une version à 3 indices, et maintenant dans un cas non orientés.

2.3.3 Programme linéaire à trois indices non-orienté

Dans cette troisième tentative, nous implémentons un PLNE à 3 indices non orienté [4] :

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e \sum_{k1}^{K} x_{ek} \\
s.c \quad & \sum_{k1}^{K} y_{ik} \ 1 \qquad \forall i \in V / \ \{0\} \\
& \sum_{k1}^{K} y_{0k} \ K
\end{aligned}$$

$$\sum_{e \in \delta(i)} x_{ek} \ 2y_{ik} \qquad \forall i \in V, k \ 1, ..., K$$
$$\sum_{i \in V} d_i y_{ik} \leqslant C \qquad \forall k \ 1, ..., K$$
$$\sum_{e \in \delta(S)} x_{ek} \geqslant 2y_{hk} \qquad \forall S \subseteq V/\{0\}, h \in S, k \ 1, ..., K$$
$$y_{ik} \in \{0,1\} \qquad \forall i \in V, k \ 1, ..., K$$
$$x_{ek} \in \{0,1\} \qquad \forall e \ \delta(0), k \ 1, ..., K$$
$$x_{ek} \in \{0,1,2\} \qquad \forall e \in \delta(0), k \ 1, ..., K$$

L'intérêt de cette formulation est direct, car dans la suite du projet, pour l'ajout des contraintes avec les compétences des techniciens, nous devrons savoir quelle voiture correspond à quelle tournée dans le but d'imposer un client à certains véhicules.
De plus, cette formulation a un avantage majeur. En effet lors de l'appel au callback ILOLAZY, il n'est pas nécessaire cette fois-ci d'ajouter des contraintes de capacité dans le cas des cycles avec dépôt. C'est la contrainte (4) qui implique le respect de la capacité de chaque véhicule. Donc comme précédemment, dans le cas d'une solution entière, nous appliquons une coupe minimum. Et dans le cas de solution fractionnaire, nous appliquons la détection de cycle sans dépôt sans faire appel cette fois-ci à la détection de violation de capacité.

Le PLNE utilisé retourne bien un solution sans sous tournées et qui respecte de bien la capacité de chaque véhicule, cependant, je me heurte à un autre problème, il ne donne pas l'optimal.

2.3.4 Évaluation expérimentale

Le PLNE à deux indices n'étant pas exploitable, seul les résultats du PLNE à 3 indices sont afficher ici.

Instance	Optimal	Temps (seconde)
P-n16-k8	450	1.13

Table 3 – Sans coupe

Remarque : Lorsqu'on enlève la coupe, on a une résolution exacte de P-n16-k8 en 1.13 seconde, mais lorsqu'on rajoute la méthode de coupe, on constate qu'aucune coupe est rajouté et pourtant celle-ci prend 172.82 seconde à trouver l'optimum.

Instance	Optimal	Temps (seconde)
P-n16-k8	450	172.82
P-n19-k2	292	0.24

Table 4 – Avec coupe

Lorsqu'on exécute le PLNE avec la méthode de coupe, le programme nous affiche bien en temps réel les cycles sans dépôt et l'ajout de la contraint d'élimination de sous tour correspondante à ce cycle, et pourtant il n'y aucune amélioration dans l'approche de la solution.

3 Le problème de tournée de techniciens

Afin de déterminer une solution heuristique du problème avec ajout de la capacité d'un technicien dans le but d'aller répondre à une demande d'un client particulier, on peut utiliser l'heuristique déterminée dans la première partie.
En effet, si on prends l'exemple donné dans l'article REFFF avec le graphe, inspiré, en Figure 34 on peut :

- ➢ Déterminer tous les problèmes de VRP (un seul vendeur pour un nombre déterminé de clients). Par exemple, en Figure 34, cela représenterais, pour le technicien A, le graphe exclusivement formé des noeuds bleus. Comme il est obligatoire pour le technicien A de visiter ces noeuds, alors on déduit de sa capacité totale la capacité de la somme des demandes de ses clients (on déduit la somme des demandes de noeuds bleus ici).
  On fait pareil pour tous les noeuds pouvant être visités par un seul technicien. Dans cet exemple, les noeuds Bleus, Jaunes et Rouges.
- ➢ Dans un second temps, avec les capacités des techniciens mises à jour (avec déductions de leurs obligations), on lance notre algorithme sur les instances pouvant avoir 2 techniciens de manière indifférente, donc les noeuds Verts dans ce cas-ci. Les 2 meilleurs sous tours sont retournés ici.
  Ensuite on déduit encore les capacités des techniciens avec l'ajout de ces nouveaux clients respectifs.
  On ré-itère sur tous les sous ensembles de techniciens que les clients rendent possible. Dans l'exemple ici, il y a simplement A et B ou A, B et C.
- ➢ Une fois que chaque sous ensemble de noeuds est déterminé pour chaque technicien alors on applique un algorithme de VRP simple.
  En effet si les noeuds 5,6,14,12 et 21 sont affectés au techniciens A alors il faut lui déterminer le moyen de plus rapide d'effectuer sa tournée.

Dans ce cas si, et surtout comme les instances reste de tailles convenables, en effet il est rare de confier 100 villes à visiter pour un technicien dans un journée par exemple, on peut résoudre ce problème de manière exacte avec un PLNE.
Ce procédé nous permet d'hybrider de la résolution par heuristique dans la détermination des sous tournées avec de la résolution exacte pour le meilleur déplacement possible d'un technicien.

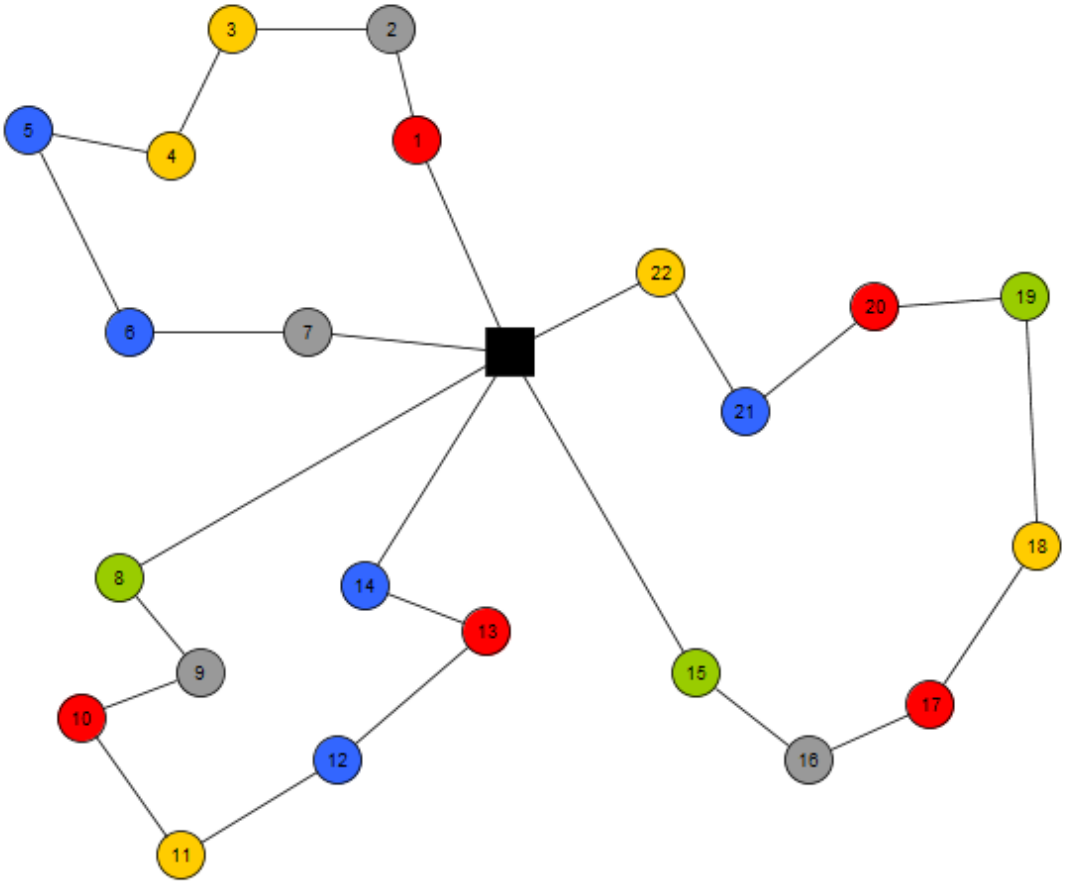

FIGURE 34 – Schéma inspiré de l'article[1] donné pour ce sujet, solution de tournée de véhicule ne respectant pas les compétences.

FIGURE 35 – Affectations possibles clients/techniciens
(A, B et C selon la couleur)

Ci-dessus on représente les sous tournées qui pourraient être trouvées mais on peut constater qu'elles ne correspondent pas aux exigences et/ou besoins des clients. En dessous on représente les tournée formés par les techniciens ayant des clients qui les veulent exclusivement. Il n'y a pas besoin de faire une résolution exacte ici car les listes des clients pour chaque technicien ne sont pas complètes.

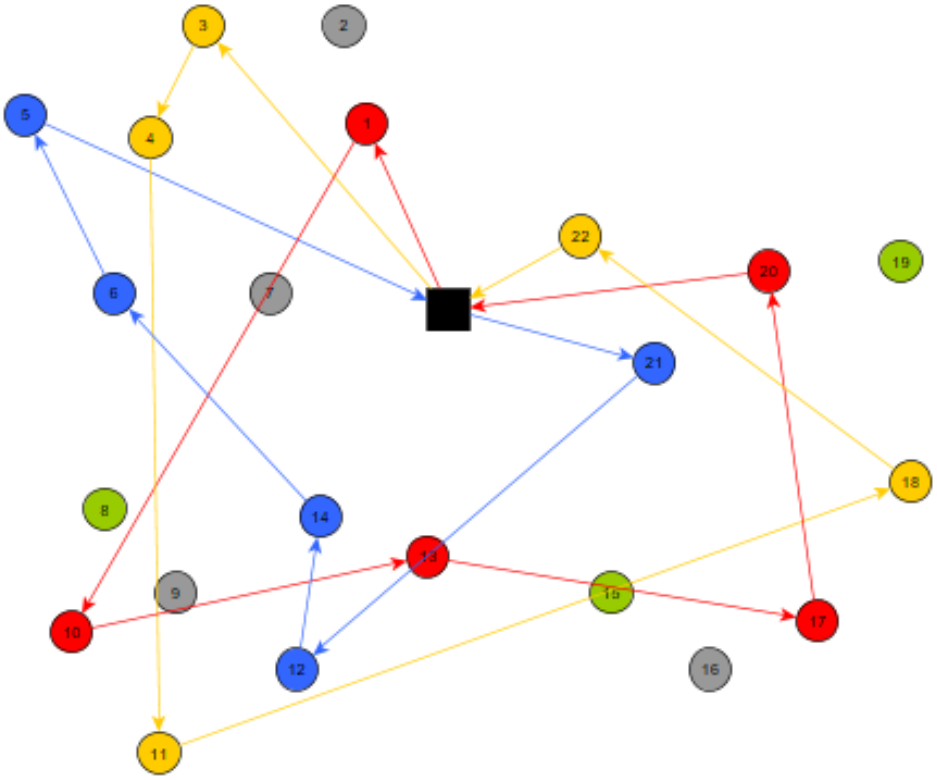

FIGURE 36 – **Étape 1** : Formation des listes de clients
obligatoires à un technicien particulier.

On constate que certains clients sont non desservis, les verts et les gris. Avec les nouvelles capacités, on lance notre heuristique sur les points Verts avec les tech-

niciens A et B et les point Gris avec les techniciens A, B et C (avec les capacités encore mises à jour). On obtient alors une liste de noeuds à visiter pour chaque technicien représenté dans le graphe ci dessous. Les noeuds verts et gris sont entourés par la couleur du techniciens (bleu pour A, jaune pour B et rouge pour C) qui les visitera finalement.

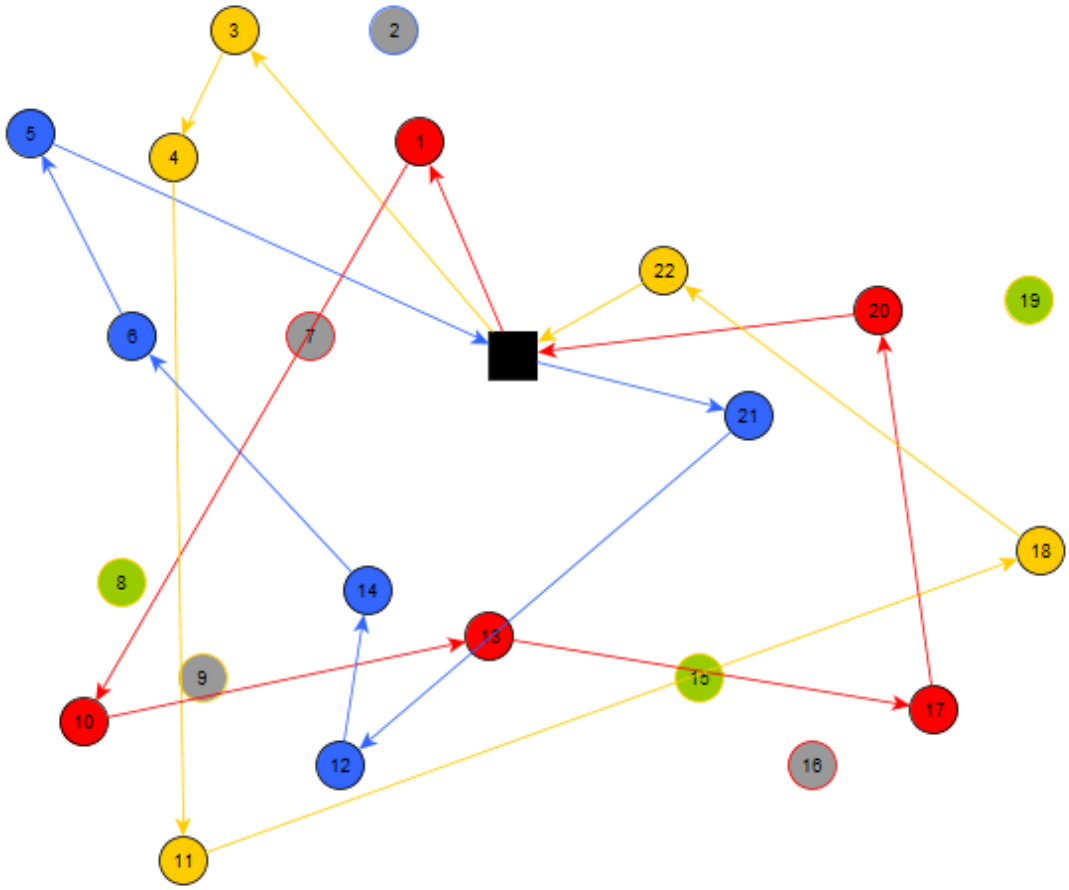

FIGURE 37 – **Étape 2** : Répartition des techniciens sur des clients moins exigent jusqu'à ce que chaque client ait un affectation.

Ensuite on applique le VRP à chaque technicien par programme linéaire en nombre entier classique afin d'avoir un optimal sur cette partie de l'heuristique. Bien sûr, si le nombre de clients visité par un technicien est trop grand on peut appliquer des coupes à ce PLNE. Cela nous renverrais donc, par exemple, les sous tournées dessinées dans le graphe ci-dessous.

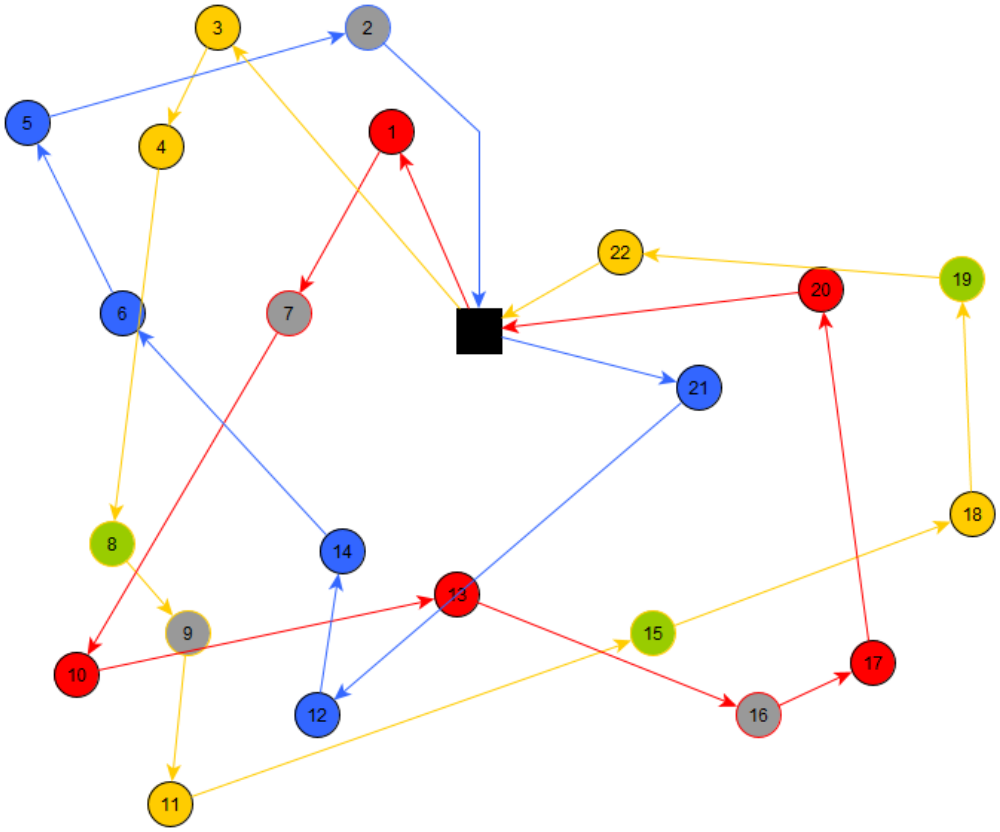

FIGURE 38 – **Étape 3** : Formation des tournées optimales pour la liste de clients par technicien trouvée de manière heuristique.

4 Conclusion

Ce projet à été très instructif sur de multiples facettes. Dans un premier temps par l'outil et le langage utilisé, en effet Cplex est le meilleur solveur disponible jusqu'à aujourd'hui et le C++ est l'un des langages les plus employé par les in-

dustries et chercheurs, notamment pour sa rapidité.
Ce projet m'a donné l'occasion de les appréhender. Aussi le sujet est un problème qui permet de se rendre compte de l'importance de l'optimisation dans le monde. En effet, presque toutes les sociétés de services et d'industries sont confrontées à ce type de problème.

Cependant, étant donné les circonstances de mes multiples programmes linéaires, il est difficile de voir une suite à ce travail autre que de faire en sorte que les PL formés fonctionne convenablement.

Annexes

A Affichage des tournées de toutes les instances exposées

Tournée A-n32-k8 avec un coût de 787.082 :

— Véhicule 1 : 22 32 20 18 14 8 27
— Véhicule 2 : 15 29 12 5 24 3 4 7
— Véhicule 3 : 19 9 10 23 16 30 11 26 6 21
— Véhicule 4 : 25 28
— Véhicule 5 : 31 17 2 13

Tournée A-n54-k7 avec un coût de 1190.94 :

— Véhicule 1 : 46 22 34 10 54 12
— Véhicule 2 : 20 9 32 41 49 38 13
— Véhicule 3 : 53 35 42 47 52 43 25 3 28 15 33
— Véhicule 4 : 7 14 23 4 45 39
— Véhicule 5 : 21 50 40 51 8 6 19 29 5
— Véhicule 6 : 36 16 11 18 27 2 37 30 24
— Véhicule 6 : 44 17 48 26 31

Tournée A-n80-k10 avec un coût de 1894.74 :

— Véhicule 1 : 2 8 22 41
— Véhicule 2 : 59 33 5 23 46 51 73 77
— Véhicule 3 : 72 15 49 19 3 38 9 25
— Véhicule 4 : 61 65 34 16 57 70 66 36 27 58 17 44 69 79 31
— Véhicule 5 : 52 78 4 75 18 7 45
— Véhicule 6 : 55 10 56 42 47 60 30
— Véhicule 7 : 40 26 48 20 62 76 21 28 32
— Véhicule 8 : 43 54 67 68 71 39 37
— Véhicule 9 : 11 53 29 80 35 12 64
— Véhicule 10 : 14 63 24 6 13 74

Tournée P-n16-k8 avec un coût de 451.335 :

— Véhicule 1 : 7
— Véhicule 2 : 3
— Véhicule 3 : 6 10 4
— Véhicule 4 : 2
— Véhicule 5 : 11 13 16
— Véhicule 6 : 14 9
— Véhicule 7 : 12 5
— Véhicule 8 : 8 15

Tournée P-n22-k2 avec un coût de 217.852 :

— Véhicule 1 : 21 6 15 18 22 8 10 14 3 7
— Véhicule 2 : 17 2 11 9 19 20 4 13 16 12 5

Tournée P-n55-k7 avec un coût de 575.258 :

— Véhicule 1 : 52 17 24 44 42 43 23 2 34 7
— Véhicule 2 : 18 41 13 27
— Véhicule 3 : 5 46 30 6 38 21 16 28 53
— Véhicule 4 : 8 54 12 39 11 32 40
— Véhicule 5 : 4 45 33 10 26 51 19 25 50
— Véhicule 6 : 47 9 36 15 20 55 14 35
— Véhicule 7 : 31 49 48 37 22 29 3

Tournée P-n101-k4 avec un coût de total 705.006 :

— Véhicule 1 : 7 95 96 97 100 60 94 86 62 17 87 39 15 45 92 101 38 99 93 98 88 14
— Véhicule 2 : 90 19 84 61 6 85 18 46 47 9 83 8 89 32 71 31 21 52 10 82 34 51 2 70 28
— Véhicule 3 : 54 59 41 3 58 43 44 16 42 23 76 75 73 74 22 5 57 24 68 40 26 56 55 25 30 69 81 13 27
— Véhicule 4 : 29 77 78 4 80 79 35 36 72 66 67 33 91 64 65 50 37 48 49 20 12 63 11 53

Tournée M-n101-k10 avec un coût de total 828.277 :

— Véhicule 1 : 44 43 42 41 45 47 46 49 52 51 53 50 48
— Véhicule 2 : 91 88 87 85 86 89 90 92
— Véhicule 3 : 99 97 96 95 93 94 98 101 100
— Véhicule 4 : 68 66 64 75 73 62 65 69 67 70
— Véhicule 5 : 11 9 10 7 8 6 4 5 3 2 76
— Véhicule 6 : 84 83 82 79 77 72 71 74 78 80 81 63
— Véhicule 7 : 21 25 26 28 30 31 29 27 24 23 22
— Véhicule 8 : 35 37 40 39 38 36 32 34 33
— Véhicule 9 : 60 61 59 57 54 55 56 58
— Véhicule 10 : 12 13 15 17 16 20 19 18 14

Tournée M- n200-k17 avec un coût de total 1486.22 :
— Véhicule 1 : 53 107 8 183 63 160 89 191 128 168
— Véhicule 2 : 52 104 162 72 66 137 36 165 35 130 78 29
— Véhicule 3 : 32 190 11 109 33 91 64 127 12 176 108 149 147
— Véhicule 4 : 14 138 88 152 100 62 114 18 174 6 119 61
— Véhicule 5 : 7 105 99 194 92 192 45 142 87 141 39 15 44 16
— Véhicule 6 : 13 55 131 166 56 26 171 68 40 180 150 27
— Véhicule 7 : 133 102 71 31 161 132 129 67 189 21 123 177
— Véhicule 8 : 69 4 170 122 30 25 164 135 151 81 178 110 196
— Véhicule 9 : 28 70 163 182 65 50 47 175 200 84 167
— Véhicule 10 : 54 41 74 172 134 198 73 22 181 106
— Véhicule 11 : 59 153 118 98 38 193 120 17 85 46 126 9 115 19
— Véhicule 12 : 157 94 86 101 143 43 173 145 58 179 116 3
— Véhicule 13 : 113 184 95 96 93 60 97 148
— Véhicule 14 : 199 111 5 156 140 188 24 187 57 76 75 23 42 146
— Véhicule 15 : 90 154 83 49 125 169 48 37 144 20 124 195
— Véhicule 16 : 112 51 103 158 34 186 159 117 197 185
— Véhicule 17 : 155 139 77 80 79 136 10 121 82 2

B Suppression du bruit des instances par critère de coût et de temps

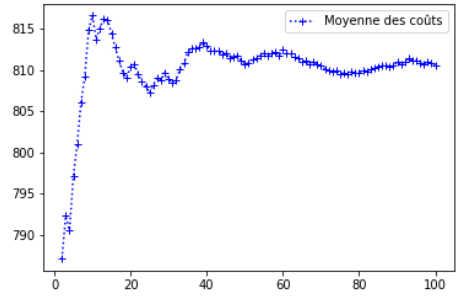

FIGURE 39 – A-n32-k8 (50,50,100)

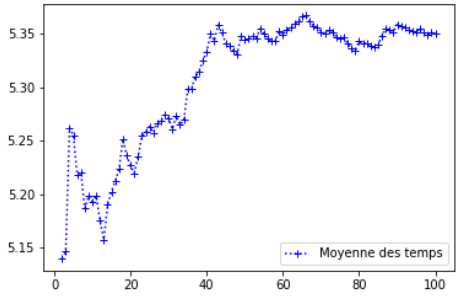

FIGURE 40 – A-n32-k8 (50,50,100)

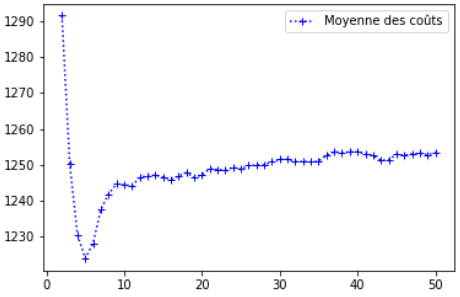

FIGURE 41 – A-n54-k7(50,50,50)

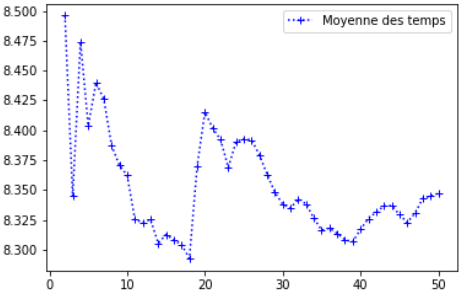

FIGURE 42 – A-n54-k7 (50,50,50)

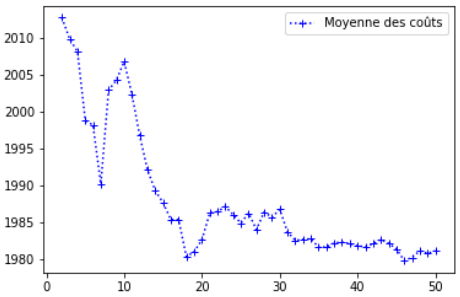

FIGURE 43 – A-n80-k10 (50,50,50)

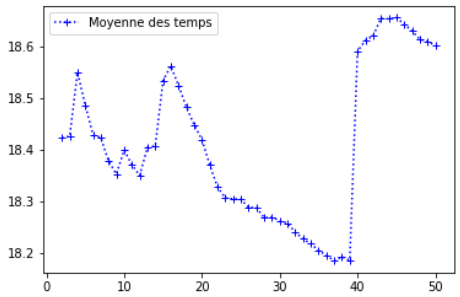

FIGURE 44 – A-n80-k10 (50,50,50)

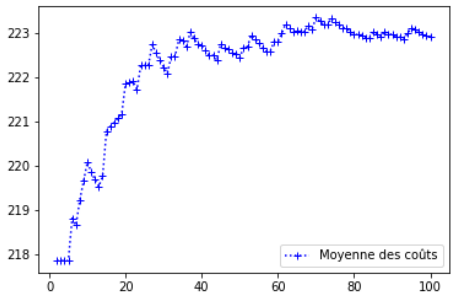

FIGURE 45 – P-n22-k2 (50,50,50)

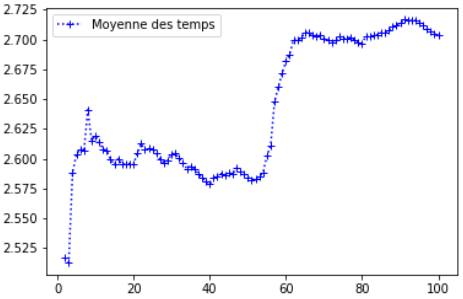

FIGURE 46 – P-n22-k2 (50,50,50)

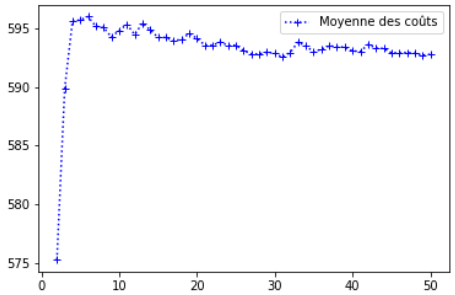

FIGURE 47 – P-n55-k7(50,50,50)

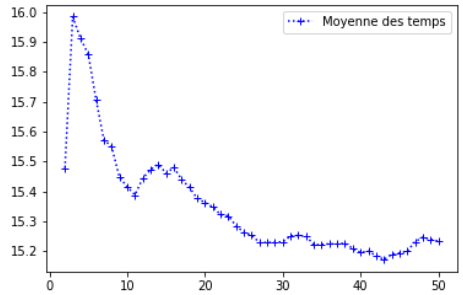

FIGURE 48 – P-n55-k7 (50,50,50)

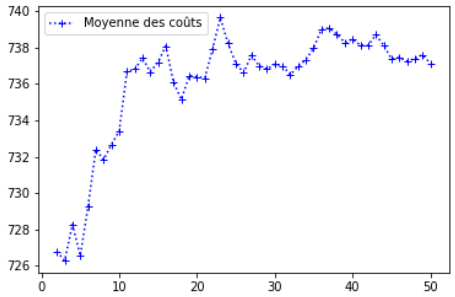

FIGURE 49 – P-n101-k4 (50,50,50)

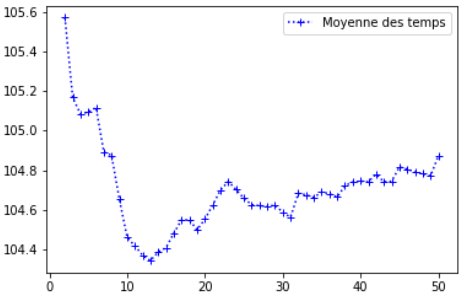

FIGURE 50 – P-n101-k4 (50,50,50)

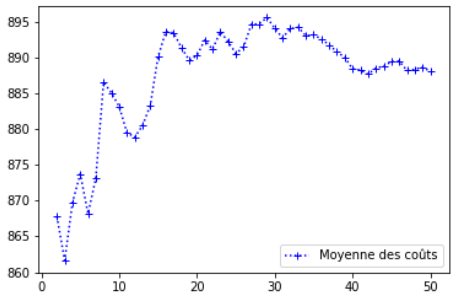

FIGURE 51 – M-n101-k10 (50,50,50)

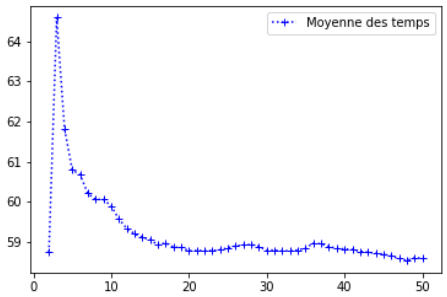

FIGURE 52 – M-n101-k10 (50,50,50)

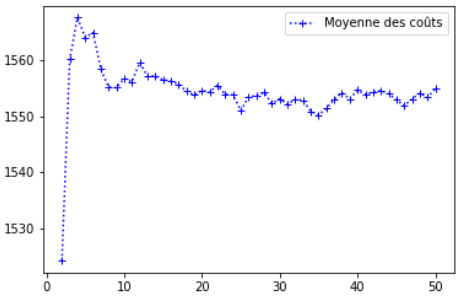

FIGURE 53 – M-n200-k17 (50,50,50)

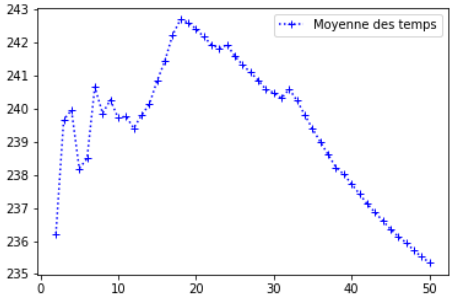

FIGURE 54 – M-n200-k17 (50,50,50)

Références

[1] Johann Dréo, Alain Pétrowski, Patrick Siarry, and Eric Taillard. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, 2003.

[2] Manfred Padberg and Giovanni Rinaldi. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming*, 47(1-3) :19–36, 1990.

[3] Philippe Augerat, José-Manuel Belenguer, Enrique Benavent, Angel Corbéran, and Denis Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2-3) :546–557, 1998.

[4] Paolo Toth and Daniele Vigo. *Vehicle routing : problems, methods, and applications*. SIAM, 2014.

[5] Jonathan F Bard, Liu Huang, Moshe Dror, and Patrick Jaillet. A branch and cut algorithm for the vrp with satellite facilities. *IIE transactions*, 30(9) :821–834, 1998.

[6] Raksmey Phan, Christophe DUHAMEL, and Philippe LACOMME. Une stratégie hybride pour le problème de tournées de véhicules avec livraisons et collectes, 2009.

[7] Shimon Even, Gene Itkis, and Sergio Rajsbaum. On mixed connectivity certificates. In *European Symposium on Algorithms*, pages 1–16. Springer, 1995.