République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLAB- BLIDA 1

Faculté des sciences

Département informatique

Mémoire pour l'obtention d'un diplôme de Master en informatique.

Option : Ingénierie du Logiciel

Thème

Optimisation des tournées d'une flotte de véhicules hétérogènes à capacités limitées avec ramassage.

Organisme d'accueil :

Centre de Développement des Technologies Avancées

Réalisé par :

- BELAIDI Boudjemaa

Présenté le 22/10/2020, devant le jury composé de :

- Mr DOUGA Yassine	Président
- Mme EL-BEY Fella	Examinatrice
- Mme LAHIANI Nesrine	Promotrice
- Mr HENTOUT Abdelfetah	Encadreur

Promotion 2019-2020

Remerciements

Avant tout je dis AL HAMDOU LI ALLAH qui me donne la volonté, la force, la patience et le courage pour réaliser ce modeste mémoire.

Je tiens à remercier toutes les personnes qui ont contribué au succès et qui m'ont aidée lors de la rédaction de ce mémoire.

Je voudrais dans un premier temps remercier, mon directeur de mémoire Madame LIAHIANI Nesrine, professeur de l'informatique à l'université de Blida, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

J'adresse mes remerciements au professeur Hentout Abdelfetah pour m'avoir accordé des entretiens et avoir répondu à mes questions sur la culture de problème de transport, ainsi que leur expérience personnelle. Il a été d'un grand soutien dans l'élaboration de ce mémoire.

Je remercie également mon collègue Abdo Ismail pour son aide à terminer de ce travail avec succès, Qu'Allah lui récompense tout le meilleur

Je remercie également toute l'équipe pédagogique de l'université de Blida Qui m'a accompagné tout au long de mes études.

Dédicaces

Je dédie ce mémoire à :

- ✓ Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

- ✓ Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

- ✓ Mes frères et sœurs qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

- ✓ Mes amis Ismail, Youcef, Haytham, Warda, Aida, Othman, Aziz, Kamel et Idriss qui ont encouragé pour ma réussite, merci à tous.

Table des matières

Résumé

L’objectif visé au départ de cette thèse était la construction d’une approche de résolution du VRP C'est de ce point de vue Nous avons donc essayé de développer un algorithme de méthode heuristique dans lequel nous avons intégré séparément la minimisation de la distance parcourue, la minimisation du temps perdu et la maximisation du taux de remplissage des véhicule. cet algorithme est alors tout désigné pour résoudre ce problème mis sous forme d’un problème de plus court chemin

Abstract

The objective at the beginning of this these is was the construction of a VRP resolution approach From this point of view, we have therefore tried to develop a heuristic method algorithm in which we have separately integrated the minimization of the distance traveled, the minimization of the lost time and the maximization of the filling rate of the vehicle. This algorithm is then ideally suited to solve this problem as a problem of shorter path.

ملخص

وكان الهدف في بداية هذه الأطروحة هو بناء نهج القرار **VRP.** من وجهة النظر هذه. حاولنا بالتالي تطوير خوارزمية طريقة ارشادية دمجنا فيها بشكل منفصل تقليل المسافة المقطوعة وتقليل الوقت الضائع وتعظيم معدل تعبئة السيارة. هذه الخوارزمية مناسبة بشكل مثالي لحل هذه المشكلة كمشكلة في مسار أقصر.

Mots-clés : Problème de tournée de véhicule, Problème de tournée de véhicule hétérogène à capacité limitée avec ramassage, résolution, optimisation, algorithmes génétiques.

LISTE DE FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

VRP: Véhicule Routing Problem.

TSP: Travelling Salesman Problem.

PVC: Problème de Voyageur de Commerce.

OC: Optimisation combinatoire.

VRP-C: Capacitated Vehicle Routing Problem.

VRP-FL: Vehicle Routing Problem with Full Truckload.

VRP-HF: Vehicle Routing Problem with Heterogeneous Fleet.

O-VRP: Open Vehicle Routing Problem.

VRP-B: Vehicle Routing Problem Back.

S-VRP: Split Delivery Vehicle Routing Problem.

VRP-PD: Vehicle Routing Problem Pick-up and Deliveries.

VRP-MD: Multi-Dépôt Véhicule Routing Problem.

VRP-1P: Vehicle Routing Problem.

MP-VRP: Problème de Tournées de Véhicule a Produits Multiples.

VRP: Problème de Tournées de Véhicule à un Seul Produit.

PVRP: Periodic Vehicle Routing Problem.

VRP-TW: Vehicle Routing Problem with Time Windows.

DVRP: Dynamic Vehicle Routing Problem.

SVRP: Stochastic Vehicle Routing Problem.

ILP: Integer Linear Programming.

UML: Unified Modeling Language.

CSP: Constraint Satisfaction Problem.

CSOP: Constraint Satisfaction Optimization Problem.

ACO: Approche colonie optimisation.

API: Application programming interface.

Introdution Générale

Les enjeux économiques et environnementaux renforcent l'importance de l'optimisation combinatoire dans les domaines de informatique et la recherche opérationnelle. C'est pour cela que de nos jours, l'optimisation des transports et la performance logistique sont des enjeux économiques très importants pour les entreprises. Cela repose notamment sur l'organisation et l'efficacité des tournées de véhicules. Ainsi, la plupart des solutions commerciales proposant une optimisation des tournées n'aboutissent pas souvent sur de véritables outils d'optimisation mais proposent plutôt, des solutions organisationnelles. Ces solutions sont souvent exploitées pour rationaliser les coûts des livraisons, des distributions ou encore des collectes.

Donc on à devant Un problème d'optimisation combinatoire, qui est un problème mathématique qui consiste a déterminer la meilleure solution parmi un ensemble fini de solutions réalisables. De nombreux problèmes font partie de cette branche,Parmi eux nous retrouvons le Problème de Tournées de Véhicules (VRP), pour lequel nous accordons une attention particulière dans le cadre de ce travail.

Nous donnons une description détaillée des composantes et des paramètres ainsi que des formulations mathématiques des Problèmes de Tournées de Vehicules classique (VRP) La présente étude est axée sur les problèmes d'optimisation combinatoire et leurs méthodes de résolution, en particulier, les problèmes de tournées de vehicules. Ces problèmes, sont qualifies de difficiles, ils appartiennent a la classe des problèmes NP-HARD[1]. La classe NP-HARD. est la plus importante des classes de problèmes d'optimisation combinatoire [2]. Des problèmes du quotidien, tels que les problèmes de distribution et ou de ramassage, les problèmes d'affectation de taches aux individus, les problèmes d'emploi du temps . . . , sont d'autres problèmes d'optimisation combinatoire de la classe des problèmes NP-HARD.

Le présent rapport, qui expose ce travail, est composé de quatre chapitres structurés

Comme suit :

1. Le premier chapitre propose des généralités sur le problème de tournée de véhicule VRP, ses fondements, son historique, son formule mathématique, et son champ d'application et la description du problème traité dans le cadre de ce travail.

2. Le deuxième chapitre présente les heuristiques et les méthodes résolution, ses paramètres, ses variantes, et la complexité, et l'approche choisir qui résoudre le problème traité dans le cadre de ce travail.
3. Le troisième chapitre la modélisation de notre application en utilisant le langage UML.
4. Le quatrième chapitre : la réalisation et l'implémentation de différentes fonctionnalités de notre programme.

Chapitre 01

PROBLEME DE TOURNEE DE VEHICULE

Introduction

En mathématiques et en économie, la théorie du transport est le nom donné à l'étude du transfert optimal de matière et à l'allocation optimale de ressources. Le problème a été formalisé par le mathématicien français Gaspard Monge en 1781 [3]. D'importants développements ont été réalisés dans ce domaine pendant la Seconde Guerre mondiale par le mathématicien et économiste russe Léonid Kantorovitch [4]. Par conséquent, le problème dans sa forme actuelle est parfois baptisé " problème du transport de Monge-Kantorovitch "[4]. Le problème de tournées de véhicules connu sous le nom de " Véhicule Routing Problem (VRP) " est une extension du problème du voyageur de commerce, connu également sous le nom de " Travelling Salesman Problem (TSP) " [5].

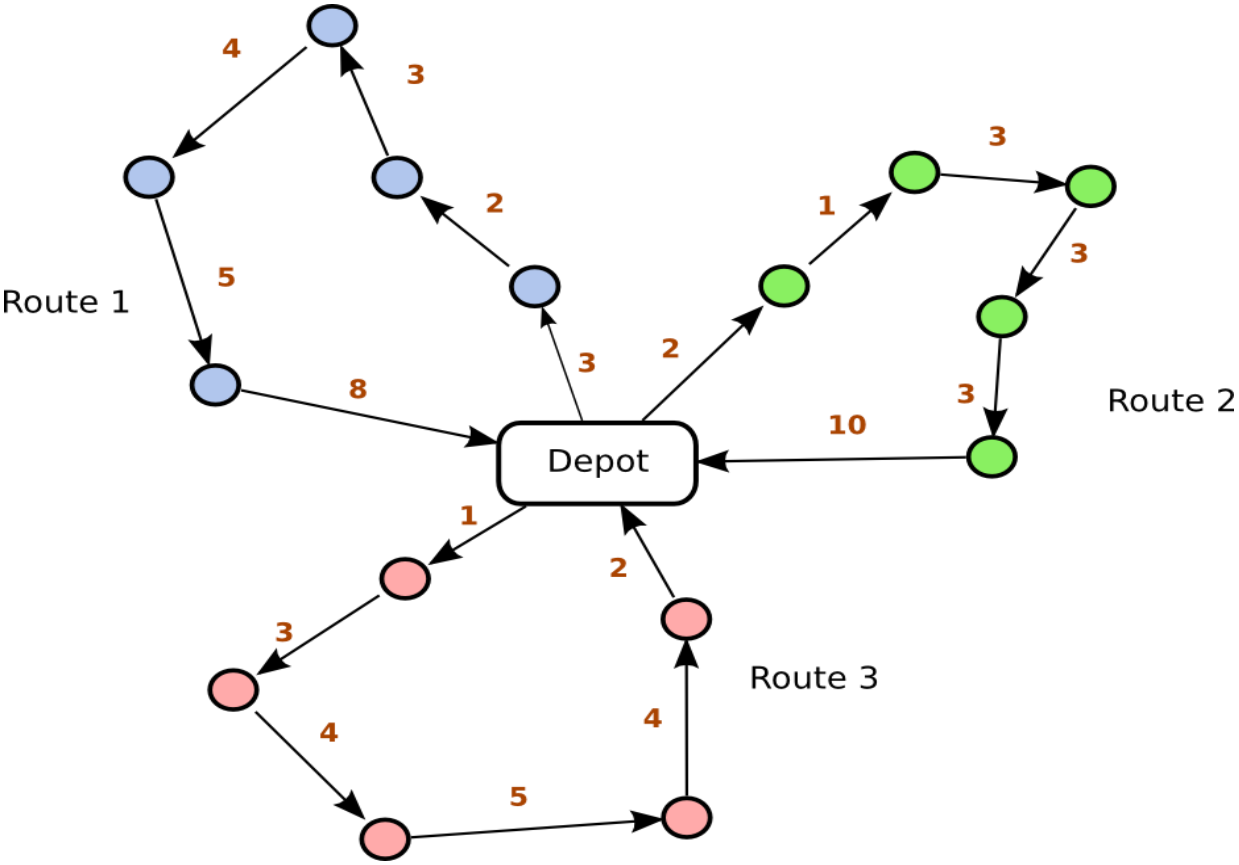


Figure 1.1 Problème de tournées de véhicules [6].

Nous allons donc dans un premier temps définir le problème du voyageur de commerce [7], puis nous verrons son extension Problème de Tournées de Véhicules.

1.1. Problème de Voyageur de Commerce(PVC)

Il est dit aussi Traveling Salesman Problem (TSP) ou PVC. C'est un problème classique de la recherche opérationnelle. Le voyageur du commerce désire visiter un certain nombre de villes (ou clients). Il doit débuter et finir sa tournée par la même ville de départ, en visitant chacune des autres villes, une et une seule fois, l'objectif étant de trouver la tournée qui minimise la distance totale parcourue. Le PVC permet la formulation de nombreuses situations réelles. La difficulté de ce problème est de trouver l'ordre dans lequel chacun des clients sera visite, en minimisant un certain critère (temps, cout du parcours, ou bien longueur totale parcourue, . . .) [7].

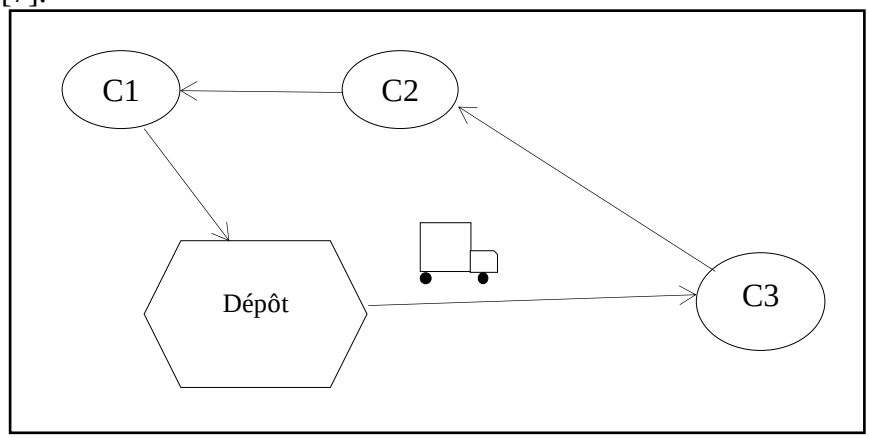


Figure 1.2 PVC

Soit G = (X, U) un graphe, où X est l'ensemble des sommets et U l'ensemble des arrêts (ou arcs si le graphe est oriente). Dantzig et al. [8], en 1954 ont proposé une formulation mathématique du PVC, ou ils définissent :

Une variable binaire Xij qui prend la valeur 1 si l'arc (i, j) est utilisé dans la tournée

Et 0 Sinon.

- Cij représente le coût du parcours de l'arc (i, j).

Ainsi nous avons la formulation suivante [9]:

$$\text{Min} \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \quad \textbf{(1.1)}$$

$$\sum_{j \in X}^{\forall j \in X} X_{ij} = 1 \quad \textbf{(1.2)}$$

$$\sum_{i \in X}^{\forall i \in X} X_{ij} = 1 \quad \textbf{(1.3)}$$

$$\sum_{i,j \in S} \leq |S| - 1; \forall S \subseteq X; 2 \leq |S| - 2 \quad \textbf{(1.4)}$$

$$Xij \in 0, 1 \forall i \in X, \forall j \in X \quad \textbf{(1.5).}$$

La contrainte **(1.1)** formule l'objectif du PVC qui est la minimisation de la longueur totale parcourue par le voyageur. Les contraintes **(1.2)** et **(1.3)** assurent que le voyageur passe une et une seule fois par chaque sommet (client). La contrainte **(1.4)** garantit l'élimination de la formation de sous-tour [10].

1.2. Problème de Tournées de Véhicules (VRP)

Le problème de tournées de véhicules (aussi appelé VRP pour *Vehicle Routing Problem*) est une classe de problèmes de recherche opérationnelle et d'optimisation combinatoire. Il s'agit de déterminer les tournées d'une flotte de véhicules afin de livrer une liste de clients, ou de réaliser des tournées d'interventions (maintenance, réparation, contrôles) ou de visites (visites médicales, commerciales, etc.). Le but est de minimiser le coût de livraison des biens. Ce problème est une extension classique du problème du voyageur de commerce, et fait partie de la classe des problèmes NP-complet.

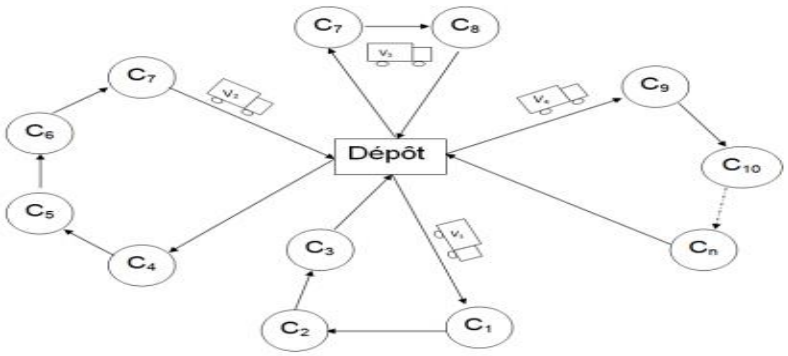


Figure 1.3 VRP.

1.3. Formulation mathématique du VRP

De nombreuses formulations du VRP existent en littérature. Le point commun de toutes ces formulations est la représentation du problème de tournées de véhicules sous forme d'un graphe orienté. La représentation graphique du VRP classique est décrite comme suit :

Soit G = (X, U) un graphe orienté où :

- X, représente l'ensemble des sommets du graphe G, il représente les clients et le(s) dépôt(s) du VRP, X = N S {0} et (|X| = n + 1).
- U, l'ensemble des arcs du graphe, il représente les chemins reliant les clients entre eux et au dépôt du VRP, (|U| = ((n + 1) n)/2).
- Une pondération peut être effectuée sur les sommets (resp. arêtes) pour définir la quantité demandée par le client (resp. distance séparant deux clients ou le temps de déplacement, . .).

La formulation mathématique du VRP classique la plus communément utilisée dans la littérature est celle adoptée par Laporte [11], Rego and Roucairol [12], Toth et Vigo [13], Crainic et Semet [14]. En effet, elle nécessite la définition de n × n variables de décision de type binaire, à trois indices, suivantes [9]:

$$X_{IJ}^{K} \begin{cases} \textbf{1} \text{ si l'arc } \textbf{(i, j)} \text{ est parcouru par } \textbf{k} \text{ éme véhicule} \\ \textbf{0} \text{ sinon} \end{cases} \qquad \textbf{(1.6)}$$

Autrement dit :

$$x_{ij}^{k} \in \{0, 1\};\ i = 0, \cdots, n\ ;\ j = 0, \cdots \sum, n\ ;\ k = 1, \cdots, m \qquad \textbf{(1.7)}$$

Le problème se modélise comme suit [9] :

Minimiser $\sum_{i=1}^{n} \sum_{i=1}^{n} Cij \sum_{k=1}^{m} x_{ij}^{k}$ **(1.8)**

Sous les contraint

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ij}^{k} = 1;\ j=2;\ldots;\ n \qquad (1.9)$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}^{k} = 1;\ i=1;\ \ldots;\ n \qquad (1.10)$$

$$\sum_{i=1}^{n} x_{ip}^{k} - \sum_{j=1}^{n} x_{pi}^{k} = 0;\ k=1;\ \ldots,\ m;\ p=1;\ \ldots:\ n \qquad (1.11)$$

$$\sum_{i=1}^{n} qi\ (\sum_{j=1}^{n} x_{ij}^{k}\ ) <= Q_k\ ;\ k = 1\ldots,\ m \qquad (1.12)$$

$$\sum_{i=1}^{n} S_i^k\ x_{ij}^{k} + \sum_{i=0}^{n}\sum_{j=0}^{n} t_{ij}^{k}\ x_{ij}^{k} <= T_k;\ k=1\ldots,\ m \qquad (1.13)$$

$$\sum_{j=1}^{n} x_{0j}^{k} <= 1;\ k=1\ldots,\ m \qquad (1.14)$$

$$\sum_{i=1}^{n} x_{i0}^{k} <= 1;\ k=1\ldots,\ m \qquad (1.15)$$

$$\sum_{i\in S}^{n}\sum_{j\in S}^{n} x_{ij}^{k} <= |S| <= (N-1) \qquad (1.16)$$

La formule (1.8) représente la fonction à optimiser ou la fonction objective du VRP classique. Généralement, l'objectif est de trouver le minimum du coût global des tournées. Les formules de (1.9) à (1.16) représentent les contraints du problème :

- Les contraintes (1.9) et (1.10) assurent que chaque client n'est servi qu'une et seule fois et par un et un seul véhicule.
- La contrainte (1.11) assure la continuité de la tournée : non seulement un véhicule doit passer une et une seule fois chez un client (1.9) et (1.10) mais il doit impérativement le quitter une fois le service est achevé.

- La contrainte (1.12) assure que la capacité du véhicule ne sera pas dépassée.
- La contrainte (1.13) assure que la durée totale d'une tournée ne dépassera jamais sa durée totale maximale.
- Les contraintes (1.14) et (1.15) assurent le non dépassement de la disponibilité de chacun des véhicules, un véhicule ne sort du dépôt et n'y revient qu'une seule fois.
- La dernière contrainte (1.16) garantit l'élimination de sous-tours.

1.4. Champs d'Application

Les applications du PVC et VRP sont aussi diverses que variées. Toute entreprise industrielle désire améliorer l'efficacité de sa chaine logistique, pour assurer une production de biens ou de services au moindre cout et une fluidité d´écoulement de sa marchandise. En effet, le problème de tournées de véhicules est un maillon principal dans le domaine de la logistique. Vu l'augmentation du cout de l'´énergie, près de 30% du cout de production d'un bien ou service sont dus aux frais de transport.

Le problème de tournées de véhicules fait partie de notre quotidien, a commencer par le ramassage scolaire, ramassage du personnel, le ramassage des dèches ménagères, la cueillette de lait cru, la distribution des journaux, de courrier et de denrées alimentaires telles que le lait, le pain, l'eau, etc. Aussi les services ambulatoires et la distribution urgente de produits pharmaceutiques font partie des problèmes de tournées de véhicules. Ces deux derniers ont la particularité de servir les clients selon la priorité de l'urgence et l'objectif n'est pas de minimiser la distance totale parcourue mais de servir le maximum de clients avant que ce ne soit trop tard.

Nous attirons l'attention sur le fait que tous ces problèmes sont contraints par le temps, et dans les problèmes de ramassage une contrainte de temps est plus rigide que pour les autres les problèmes de distribution.

L'informatique et la robotique sont d'autres domaines ou le VRP est largement utilise. En effet, le routage de l'information dans les réseaux informatiques à connexion distante ou par câble est fondé essentiellement sur le VRP. La planification du calendrier aérien est une autre application du VRP.

En dépit de l'intérêt que porte l'homme à ce problème, à ce jour, il se trouve incapable de résoudre avec exactitude certaines de ses instances. Les grandes instances du VRP s'avèrent difficiles ou complexes, parfois même impossibles, a résoudre avec un calculateur quelle que soit la puissance de celui-ci.

1.5. Paramètres du VRP

Le VRP est caractérisé par :

1.5.1. Réseau

Le réseau de transport permet la circulation des flux d'individus, de fret ou des informations. Il est en quelque sorte le squelette d'un système visant à établir une forme de communication. Il peut être schématisé sous la forme d'un graphe complet, symétrique ou asymétrique. Les sommets représentent les clients, caractérises par sa position géographique (les coordonnées) et les arêtes représentent les chemins reliant les différents clients. Sous certaines contraintes, ce graphe peut être oriente.

1.5.2. Clientèle

Le client est caractérisé par sa demande qui peut être une demande d'un service ou de produits (marchandises), ces produits peuvent être d'un seul ou de plusieurs types.

La demande totale des clients d'une même tournée ne doit pas excéder la capacité Q du véhicule. Aussi, il est caractérisé par sa position dans l'espace. Enfin la demande peut être déterministe (quantité demandée par le client est fixe et connue par le distributeur) ou incertaine (stochastique).

1.5.3. Flotte de véhicules

Le premier critère de la flotte est sa taille (le nombre de véhicules la composant), le second est son homogénéité (les véhicules sont caractérises par la même capacité d'emport et le même cout de transport) ou hétérogénéité (les véhicules ont des capacités d'emport et/ou couts de transport différents).

1.5.4. Fonction objectif

Les objectifs les plus courants sont soit la minimisation soit Maximisation :

- Minimisation du temps total du parcours de la tournée, du temps d'attente, du temps du retard, du temps de service, . . .
- Minimisation du nombre de véhicules.
- Minimisation du cout total de la tournée, cout fixe à savoir l'amortissement du matériel (véhicule ou autre), salaire des chauffeurs, frais des véhicules, . . .
- Maximisation du gain engendré par la tournée dans le cas de collecte de produits chez des clients.
- Maximisation de la qualité de service.
- Maximisation du chargement des véhicules utilisés pour les tournées.

1.6 Description du problème traité dans le cadre de ce travail

Le problème de tournées de véhicules (Vehicle Routing Problem, VRP) est un domaine de recherche très connu en recherche opérationnelle. Ses différentes variantes ont été largement explorées dans la littérature. Néanmoins, la recherche autour de cette problématique est toujours d'actualité. Le problème abordé dans le cadre de ce sujet de Master est une variante de ce problème. Il est défini par la donnée de **K** véhicules à capacités différentes **Qi (i ≤ K),** où chacun peut effectuer les tâches de ramassage au niveau d'un ensemble de m sites de ramassage parmi les n sites existants **(m ≤ n).** Chaque véhicule **Vi (i ≤ K)** doit donc visiter un ensemble de sites dont la somme des quantités à ramasser ne doit pas dépasser sa capacité **Qi (i ≤ K),** avant de transporter ce qui a été ramassé à un emplacement final (dépôt, décharge, etc.). Chaque site doit être visité par un seul véhicule qui ramasse, en une seule fois, complétement son contenu. Les sites sont localisés en des emplacements géographiques différents où la distance entre chaque deux site est connue a priori. Une solution à ce problème consiste à définir pour chaque véhicule une tournée, de manière à minimiser la distance totale parcourue par tous les véhicules tout en respectant les contraintes imposées.

Conclusion

Dans ce chapitre nous avons présenté le problème de la tournée de véhicule. Comme nous l'avant déjà ce problème appartient à la classe NP difficiles. Donc sa résolution n'est possible que d'une manière approchée car de nos jours il n'y a aucun algorithme qui puisse nous permettre leur résolution en un temps polynomial. Alors dans le prochain chapitre, nous introduisons l'approche adaptée pour résoudre ce problème et nous allons voir en détail le fonctionnement d'une solution par optimisation de tournée de véhicules à capacité limitée avec ramassage**.**

Chapitre 02

METHODES DE RESOLUTIONS

Introduction

Ce chapitre est consacré à la présentation de principales approches de résolution de problèmes d'optimisation combinatoire de type NP-HARD :

Nous commençons par la présentation des approches exactes, en citant quelques résultats célèbres, ensuite, nous passons aux approches heuristiques et aux méta-heuristiques et nous donnons une description générale des approches heuristiques et méta-heuristiques et nous présentons leurs concepts communs. Puis, nous distinguons les différents types d'approches selon le nombre de solutions manipulées à chaque itération (S-Méta-heuristique et P-Méta-heuristique), la stratégie de recherche (méthodes de voisinages) ou la source d'inspiration (colonie de fourmis, . . .). Nous terminons par une conclusion sur les avantages et inconvénients de chacune des approches.

2.1 Approches exactes

Les méthodes exactes, dites aussi complètes, assurent la détermination de solutions optimales pour les problèmes d'optimisation combinatoire, suite à l'exploration exhaustive par énumération de l'ensemble des solutions réalisables de ces problèmes.

Durant les soixante dernières années, une large gamme de méthodes exactes a et conçue pour la résolution des problèmes d'optimisation combinatoire en général et particulièrement, les problèmes du VRP, qui ne cessent d'attirer l'attention des chercheurs.

Parmi les aboutissements les plus récurrents nous citons : Programmation Linéaire, la programmation Dynamique et les approches Branch and Bound.

2.1.1. Programmation Linéaire

C'est une approche dédiée exclusivement aux programmes linéaires de type :

$$(P) = \begin{cases} \textbf{Min } Z = C.x \\ \textbf{S.C.} \\ A.x = b \;\; ; \;\; x >= 0 \end{cases}$$

Avec :

- **n** : nombre de variables **(x = (x1, · · ·, xn)) ;**
- **m** : nombre de contraintes **m _n ;**
- **A = (aij) ; i = 1, · · ·, m; j = 1, · · ·, n :** matrice des contraintes **(m×n)** et rang**(A) = m;**
- **C = (C1, · · ·, Cn) :** vecteur ligne des profits (ou gains) ;
- **b = (b1, · · ·, bm) :** vecteur colonne des seconds membres.

La résolution de ce problème consiste à affecter des valeurs au vecteur **x = (x1, · ·, xn)** de sorte que la fonction **Z** soit à son minimum, sans violation de la moindre contrainte.

En 1947 G. Dantzig, a développé la méthode du simplexe [15]. C'est la méthode de résolution de programmes linéaires la plus utilisée à nos jours. Elle s'avère pratique, simple et efficace jusqu'à ce que Klee et Minty [16], en 1972, mettent en évidence des exemples pour lesquels la complexité de l'algorithme du Simplexe croit exponentiellement avec la taille du problème à résoudre, raison pour laquelle les chercheurs se sont remis à explorer de nouvelles pistes de la recherche de méthodes de résolution efficaces.

Les explorations les plus fructueuses sont celles qui ont abouti à la méthode des ellipsoïdes élaborée par Khachian [17]. Cette méthode consiste à utiliser une suite d'ellipsoïdes de volumes décroissants mais contenant à chaque itération la solution optimale du programme linéaire à résoudre. La méthode du point intérieur développée par Karmakar [18], basée sur le principe de recherche de points à l'intérieur du polyèdre des contraintes permettant le passage rapide au sommet optimal. Parmi les dérivées de cette dernière, on cite la méthode de barrière qui se trouve plus efficace que la méthode du simplexe pour certains problèmes de grande taille [19]. Il y a aussi l'algorithme de génération de colonnes qui est une technique de résolution exacte très sophistiquée, appliquée efficacement à des problèmes de grandes instances [20]. Ces méthodes ne sont pas appropriées aux programmes linéaires en nombres entiers, mais peuvent être utilisées dans des schémas de résolution bases sur le principe de l'arborescence et celui des coupes.

2.1.2. Programmation Dynamique

C'est une méthode basée sur le principe de Richard Bellman (1957), qui stipule que " une sous-stratégie d'une stratégie optimale est elle-même optimale " [21]. Eilon, Wastson-Gandy et

Christofides [22], ont appliqué ce principe pour la résolution du VRP. Ils l'ont subdivisé en sous-problèmes simples et faciles à résoudre. Les sous problèmes obtenus sont de taille réduite et peuvent être traités. Ils sont liés au problème principal par des contraintes supplémentaires. Leur résolution séquentielle suivie d'une remontée graduelle au problème principal assure la détermination de la solution du problème principal.

Le point fort de cette méthode est le fait d'éviter d'évaluer deux fois la même fonction, généralement en utilisant une table de résultats obtenus, remplie à fur et à mesure que l'on résout les sous-problèmes et c'est ainsi qu'on gagne un temps considérable.

2.1.3. Approche de Branch and Bound

L'approche de Branch and Bound permet une exploration intégrale du domaine des solutions réalisables d'une façon arborescente, où est la racine de l'arbre et ses feuilles sont des sous-ensembles de S qui seront à leur tour des racines de nouvelles feuilles.

Dans l'approche de Branch and Bound, on distingue deux étapes principales : la séparation et l'évaluation.

L'étape de séparation permet le passage de la racine à la feuille. Elle partitionne progressivement les ensembles non stériles des solutions en sous-ensembles de taille réduite tout en formant un recouvrement de l'ensemble de solutions S, ensuite on évalue chacun des sous-ensembles obtenus en leurs associant la valeur du coût des solutions leurs appartenant. Le parcours de l'arborescence se fait de trois manières différentes :

- En largeur, le sommet à séparer est le premier créé.
- En profondeur, le sommet à séparer est le dernier créé.
- En choisissant la stratégie du meilleur d'abord.

La stratégie du parcours joue un rôle déterminant sur l'efficacité de l'algorithme. En pratique la stratégie du parcours en profondeur est la stratégie la plus répandue. En effet, elle permet de trouver très rapidement les solutions réalisables dont l'évaluation affine la borne supérieure et permet de couper plus vite les branches stériles.

L'injection de l'algorithme de génération de colonnes dans l'algorithme de Branch and Bound génère l'algorithme de Branch and Price, qui s'avère efficace pour les problèmes du VRP d'instances dépassant 50 villes [23].

Christofied, Mingozzi et Toth [24], ont appliqué la méthode de séparation et évaluation au VRP à contraintes de capacités de véhicules variables. La meilleure instance résolue est de 53 clients et 08 véhicules. Après résolution du problème, le nombre maximal de véhicules dans une tournée est 15.

2.1.4. Avantages et Inconvénients

Les méthodes exactes garantissent l'optimalité des solutions qu'elles fournissent. En effet, la détermination de la solution optimale se fait par un parcours exhaustif du domaine des solutions réalisables du problème.

Pour les problèmes d'optimisation combinatoire, la taille du domaine des solutions réalisables d'un problème croient exponentiellement avec sa taille; donc, le parcours exhaustif ce domaine des solutions réalisables est impossible à partir d'une certaine taille du problème.

2.2. Approches Heuristiques et Méta-Heuristiques

La majorité des problèmes d'optimisation combinatoire font partie de la classe NP-HARD, y compris le VRP abordé dans le chapitre précédent. La complexité des problèmes rend les méthodes de résolution exactes inefficaces, et vu la fréquence et récurrence de ces problèmes en pratique, les chercheurs ont été menés à développer des méthodes approchées pour résoudre ces problèmes complexes. En effet, les méthodes approchées donnent des solutions appréciables à des problèmes de complexité importante en un temps raisonnable.

2.2.1. Heuristiques

Le terme heuristique vient du verbe grec "heuriskein" signifiant "trouver". Une heuristique permet de trouver une "bonne" solution, en un temps raisonnable, seulement elle n'offre aucune garantie sur l'optimalité de la solution trouvée.

Selon Feignebaum et Feldman [25], une heuristique est une règle d'estimation, une stratégie, une astuce ou bien une simplification, qui limite la recherche de bonnes solutions dans l'espace des configurations de façon ahurissante.

La performance d'une heuristique apparait essentiellement, en sa complexité spatiale et temporelle, et en sa simplicité et facilité d'implémentation.

Sa flexibilité et son débit de génération de solutions sont aussi des critères de performance d'une approche heuristique. Néanmoins, il est impossible d'évaluer théoriquement avec exactitude les performances d'une heuristique.

2.2.2. Méta-Heuristiques

"Méta" est un préfixe signifiant "au-delà", "plus que ça", "à un niveau supérieur", lorsqu'il est rajouté à l'heuristique il donne "Meta-Heuristique" qui veut dire "trouver au-delà", ou "trouver plus que ça". Avant l'apparition du terme méta-heuristique, on parlait plutôt des heuristiques modernes.

Les méta-heuristiques sont des stratégies d'exploration du domaine de solutions réalisables à la recherche de la solution "optimale" ou presque. Ce sont des approches guides intelligemment par l'usage de simples procédures de recherche locale et/ou des processus d'apprentissage complexes de sorte à éviter le bouclage ou les optimums locaux. Elles sont d'ordre général, s'adaptent à tous types de problèmes sans perte d'efficacité. Comme les heuristiques, les méta-Heuristiques n'offrent pas de garantie sur l'optimalité, bien qu'on ait pu démontrer la convergence de certaines d'entre elles vers l'optimum global.

Non déterministes, elles incorporent souvent un principe stochastique pour surmonter l'explosion combinatoire.

2.2.3. Critère d'Arrêt

Comme tout processus itératif, les méta-heuristiques sont dotées d'un critère d'arrêt. Il peut être une limitation du nombre d'itération, l'atteinte d'un certain taux d'amélioration d'un paramètre ou de fonction. Il assure la finitude du processus de recherche et lui évite de tourner indéfiniment. Le critère d'arrêt se défini de deux manières :

Statique : Fixer préalablement le nombre d'itérations à effectuer. Généralement la détermination de ce nombre se fait sur la base des résultats empiriques. En algorithmique, cela se traduit par l'insertion d'un compteur.

Dynamique : Varie selon une certaine formule qui dépend des résultats des itérations précédentes d'une fonction aléatoire.

2.3. Complexité algorithmique

La complexité d'un problème quelconque sous-entend l'estimation, dans le pire des cas, du nombre d'instructions à effectuer afin de résoudre une instance du problème. La complexité algorithmique sert à mesurer l'efficacité d'un algorithme de résolution d'un problème. En effet, plus le nombre d'instructions est important plus le problème est complexe et donc la complexité algorithmique est proportionnelle au nombre d'instructions du problème. On en déduit alors que plus l'algorithme est complexe moins il est efficace.

Plusieurs types de complexité peuvent être distingues, on cite la complexité constante qui est indépendante de la taille du problème, la complexité logarithmique, la complexité linéaire, la complexité quadratique, la complexité cubique, la complexité polynomiale, la complexité exponentielle, et enfin la complexité factorielle.

Selon le critère de complexité, nous distinguons essentiellement quatre classes de problèmes :

2.3.1. Classe P

Dans cette classe on trouve l'ensemble des problèmes pouvant être résolus, en temps polynomial, par une machine de Turing. La résolution de n'importe quelle instance des problèmes de cette classe se fait en un temps et espace polynomiale d'où l'appellation de classe de problèmes polynomiaux ou la Classe P.

2.3.2. Classe P-Complet

Dans cette classe, on trouve l'ensemble des problèmes pouvant être résolus, en temps polynomial, par une machine de Turing. La résolution de n'importe quelle instance des problèmes de cette classe se fait en un temps et espace polynomiale d'où l'appellation de classe de problèmes polynomiaux ou la Classe P.

2.3.3. Classe NP

Dans cette classe, on trouve l'ensemble des problèmes polynomiaux et l'ensemble des problèmes non polynomiaux où tout problème polynomial peut se réduire en un temps poly-logarithmique avec ou sans l'utilisation d'une machine de calcul déterministe.

2.3.4. Classe NP-HARD

Cette classe est dite aussi NP-Difficile. Elle regroupe l'ensemble des problèmes dont la complexité est exponentielle ou doublement exponentielle. Cette classe de problème se distingue par la difficulté de résolution optimale ou exacte des grandes instances, en un temps polynomial. A cet effet, on se contente de trouver des solutions dites de bonne qualité en un temps polynomial.

Parmi les problèmes de cette classe, on cite les problèmes de partitionnement d'arbres, des connexités dans un graphe de routage, ... Le problème de tournées de véhicules est un des problèmes de routage. Lenstra et Rinnooy [27] ont prouvé que le VRP est un problème NP-HARD.

Le problème de tournées de véhicules est un problème NP-HARD. Sa résolution par une méthode exacte s'avère inappropriée pour les instances de grande taille. Il est donc inévitable de procéder à sa résolution par des approches heuristiques, qui fournissent des solutions réalisables et appréciables en un temps raisonnable.

2.4 La solution proposée du problème traité dans le cadre de ce travail

Nous allons voir en détail le fonctionnement d'une solution par optimisation de tournée de véhicules à capacité limitée avec ramassage en utilisent deux méthodes (optimisation de colonie de fourmis et optimisation linéaire en nombres entiers). L'algorithme qui est montré ici est celui qui a été implémenté dans le cadre de ce travail et sera utilisé lors de la collaboration entre solutions. Il existe beaucoup de variantes de cet algorithme dans la littérature et cette méthode est appliquée pour de nombreux autres problèmes [28].

2.4.1 Optimisation de colonie de fourmis

Ce type de solution est inspiré, comme son nom l'indique, du comportement des fourmis en société. En effet, en observant leur comportement, les chercheurs ont pu constater qu'elles avaient tendance à utiliser le plus court chemin disponible à leur source de nourriture et ce malgré leurs capacités cognitives limitées. Cette intelligence collective est expliquée comme suit :

- les fourmis éclaireuses cherchent des sources de nourriture en se déplaçant aléatoirement
- après avoir trouvé de la nourriture, elles rentrent au nid en laissant derrière elle une piste de phéromones
- les autres fourmis étant attirées par les phéromones vont suivre approximativement la piste et laisser à leur tour des phéromones lors de leur retour au nid
- si plusieurs pistes de phéromones sont disponibles, les pistes les plus courtes aurone tendance à être privilégiées car elles pourront être parcourues plus de fois en un temps donné
- les autres pistes de phéromones plus longues auront tendance à disparaître car de moins en moins de fourmis l'emprunteront

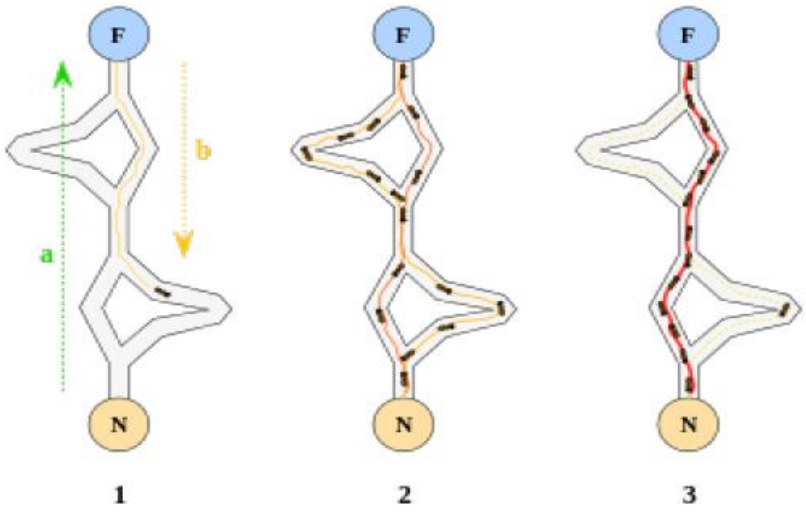


Figure 2.1 Exemple de l'intelligence collective des fourmis leur permettant de trouver le chemin le plus court à une source de nourriture [29].

Sur cette figure, la première image représente la fourmi cherchant la nourriture F (chemin a) et laissant une piste de phéromones à son retour au nid N (chemin b). La deuxième image montre le renforcement des pistes par les autres fourmis. La dernière image montre la piste finale utilisée par la colonie de fourmis.

2.4.1.1 Algorithme d'optimisation de colonie de fourmis

Cette solution utilise un processus itératif : à chaque itération de l'algorithme, on produit une nouvelle solution et on met à jour les quantités de phéromones. Quand le temps d'exécution ou le nombre d'itérations limite est atteint, on retourne la meilleure solution trouvée. On peut représenter le schéma d'exécution comme ceci :

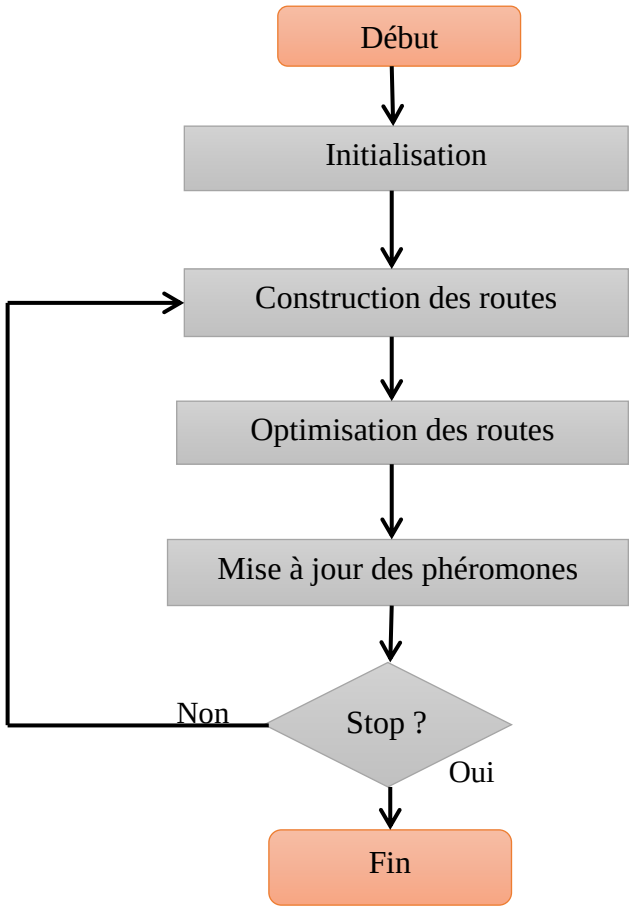


Figure 2.2 Schéma global de l'exécution de la solution.

2.4.1.2 Initialisation

Dans cette étape, on s'intéresse à l'initialisation de la quantité de phéromones présentes sur les arêtes. On peut décider de mettre une quantité de phéromones égale sur chaque arête afin de laisser les fourmis découvrir par elles-mêmes les meilleurs chemins à emprunter ou alors d'utiliser un algorithme pour déposer des phéromones sur les chemins prometteurs. Pour ce faire, on peut par exemple utiliser une des méthodes constructives vues au point 3.1.1 et déposer les phéromones sur les routes trouvées par cet algorithme. Ici, c'est l'heuristique de Clarke et Wright [30] qui est utilisée, les phéromones sont déposées exactement de la même manière que si les routes avaient été

générées par la colonie de fourmis. Ceci sera détaillé dans la partie consacrée à la mise à jour des phéromones.

2.4.1.3 Construction des routes

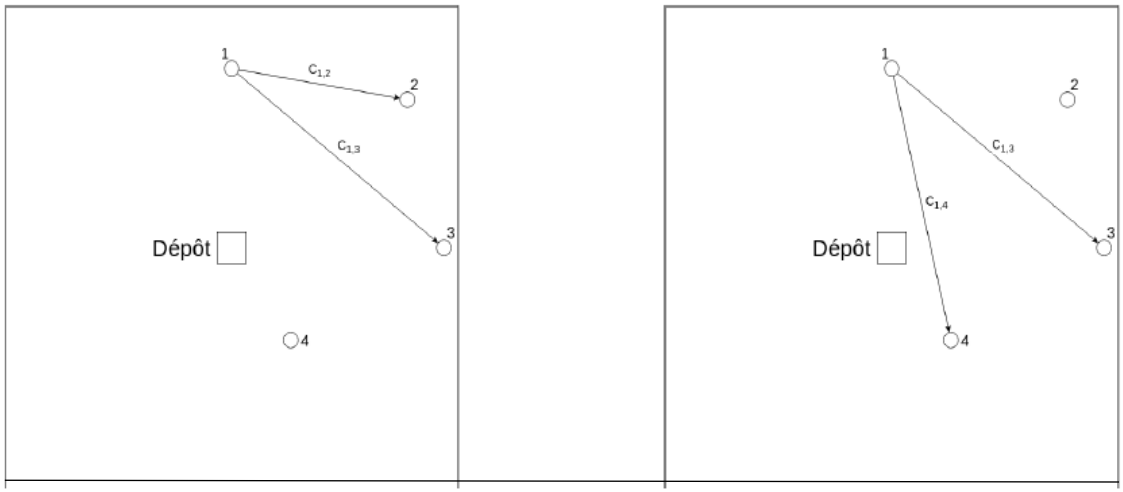
Pour construire les routes, on considère que chaque véhicule est représenté par une fourmi. On construit les routes de chaque fourmi itérativement, elle commence avec une route vide et décide à chaque itération le prochain site à desservir ou de retourner au dépôt.
Pour choisir son prochain site, la fourmi se base sur deux informations : les pistes de phéromones et la visibilité. Ces deux notions sont définies entre chaque paire de nœuds (par "nœuds", on entend un site ou le dépôt). Les phéromones doivent indiquer les routes empruntées précédemment qui ont mené à une bonne solution. La visibilité sert à contraindre la fourmi à choisir les sites les plus proches d'elle.
La visibilité peut se définir de beaucoup de manières différentes et a un impact significatif sur les performances. En principe, elle doit être plus élevée pour les sites étant de bons candidats pour la prochaine destination d'un véhicule. Le but est donc de favoriser les sites dont le coût pour les atteindre est faible et étant bien situés. Dans l'implémentation de cette méthode, c'est la définition qui est utilisée [31]:

$$nij = ci0 + c0j - g \cdot cij + f \cdot |ci0 - c0j| \qquad (2.1)$$

- ***nij*** est la visibilité entre les nœuds ***i*** et ***j***
- ***cij*** est le coût entre les nœuds ***i*** et ***j***
- ***g*** et ***f*** sont des paramètres positifs à fixer.

Les paramètres ***g*** et ***f*** ont ici été fixés à $f = g = 2$ car ce sont les valeurs ayant les meilleures performances.

Pour comprendre l'intuition derrière cette formule, supposons que le coût ***cij*** entre les nœuds ***i*** et ***j*** soit défini par la distance qui les sépare. Dans ce cas, on souhaite favoriser les sites étant proches et étant bien situés par rapport au dépôt. Un site bien situé est un qui ne force pas le véhicule à passer près du dépôt. En effet, en passant près du dépôt, on fait en quelque sorte un détour, puisqu'il faudra plus tard y terminer la tournée du véhicule. Il est donc généralement préférable que ce site soit desservi par un autre véhicule.
Voici des exemples pour illustrer ces deux aspects :



(a) En partant du site **1**, le site **2** est un meilleur candidat que le site **3** car il est plus proche.

(b) En partant du site **1**, le site **3** est un meilleur candidat que le site **4** car il est mieux situé.

Figure 2.3 Illustration de l'intuition de la formule de visibilité 2.1.

Le choix du prochain site **j** en se trouvant en **i** est fait selon la formule probabiliste suivante :

$$P_{ij} = \begin{cases} \dfrac{T_{ij}^{\alpha} \cdot n_{ij}^{\beta}}{\sum_j T_{ij}^{\alpha} \cdot n_{ij}^{\beta}} & \text{Si } \mathbf{j} \text{ est un candidat} \\ 0 & \text{sinon} \end{cases} \qquad \textbf{(2.2)}$$

T_{ij} est la quantité de phéromones sur l'arête **(i, j)**, n_{ij} est la visibilité comme définie précédemment, **α** et **β** sont des paramètres permettant d'influencer la pondération entre les pistes de phéromones et la visibilité.

Un site **j** est candidat à trois conditions :

- il fait partie des nœuds ayant la meilleure visibilité depuis le nœud actuel **i**
- la demande de ce site ne fait pas excéder la capacité du véhicule
- le site n'a pas encore été desservi.

Cette formule probabiliste vise donc à favoriser d'une part les arêtes dont la visibilité est élevée, qui fait donc du site un bon candidat, et à la fois les arêtes ayant une grande quantité de phéromones, qui ont donc déjà contribué à l'élaboration de bonnes routes dans de précédentes itérations.

En pratique, on fixe une taille limite à la liste de candidats considérés à chaque site, par exemple, on décide de ne garder au maximum que **25%** des voisins de chaque nœud. C'est un paramètre à fixer par l'utilisateur. Il a deux conséquences directes sur les performances :

- une réduction du temps de calcul pour une itération, car on considère moins de voisins
- une amélioration globale de la qualité des solutions trouvées pour un nombre d'itérations fixe, car lorsque la fourmi choisit un site lointain, cela mène généralement à une solution de moins bonne qualité.

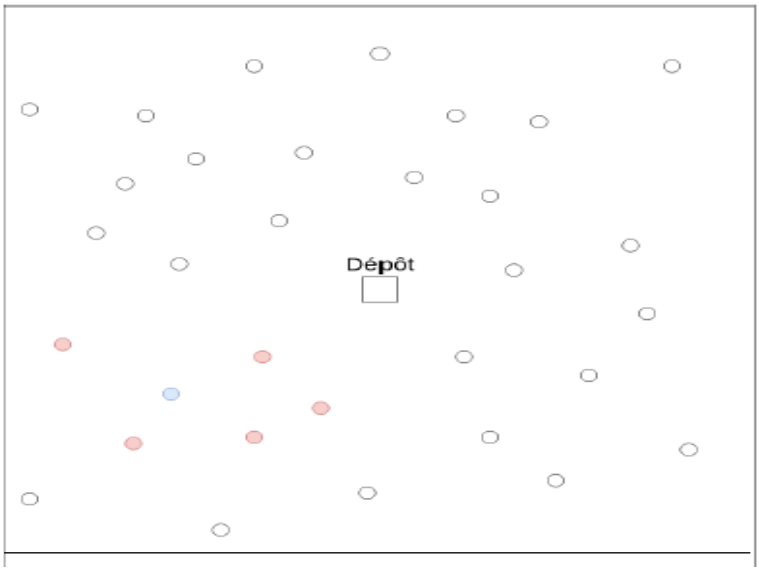


Figure 2.4 Exemple de liste de candidats. Les points rouges sont les candidats considérés en partant du site bleu.

La fourmi rentre au dépôt lorsqu'il n'y a plus aucun candidat au site où elle se trouve. On commence alors la construction de la route de la fourmi suivante de la même manière et ce jusqu'à avoir atteint le nombre de véhicules requis (donnée reçue en entrée). Il est possible que la solution trouvée n'utilise pas exactement **K** véhicules, par exemple lorsque les fourmis sont rentrées au dépôt alors qu'il leur restait un peu de capacité. Dans ce cas, l'algorithme efface les routes générées et recommence la construction des routes depuis le début.

2.4.1.4 Optimisation des routes

Cette étape vise à améliorer les routes construites en optimisant chacune des routes individuellement. Chaque route est considérée comme un TSP, le but est donc de trouver un itinéraire de coût inférieur passant par les clients assignés à ce véhicule. Pour ce faire, on utilise ici l'algorithme de recherche locale **2-opt**, il a été présenté comme suit :

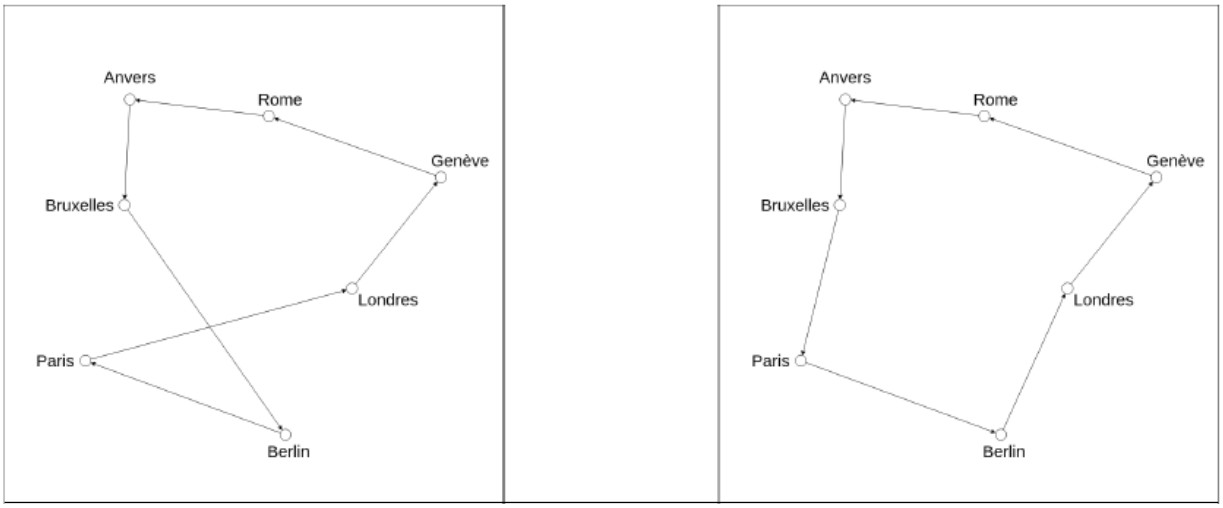
- La recherche locale est une des méthodes les plus courantes pour résoudre approximativement les problèmes d'optimisation difficiles.

- Elle se base sur des modifications successives d'une solution connue pour trouver de meilleures solutions.
- Nous allons voir ici un des algorithmes basiques utilisant la recherche locale pour trouver une solution approchée du TSP : l'algorithme **2-opt**. On se servira de celui-ci pour illustrer les notions liées à la recherche locale.
- Toutes ces notions sont bien sûrs identiques et directement transposables pour le VRP, le TSP est utilisé uniquement pour faciliter les exemples.

C'est un algorithme itératif qui cherche à chaque étape les deux meilleures arêtes à supprimer et reconnecter de manière croisée. Par "meilleure", on parle en termes de réduction du coût total de l'itinéraire. Cette déconnexion/reconnexion s'effectue comme ceci :

- on choisit deux arêtes à supprimer, notons les **(A, B)** pour l'arête allant de **A** vers **B** et **(C, D)** pour l'arête allant de **C** vers **D**.
- on reconnecte **A** à **C** et **B** à **D** et on inverse le sens des arêtes se trouvant sur le chemin de **B** à **C**.

Les deux arêtes qui sont déconnectées/reconnectées ne peuvent pas avoir d'extrémités en commun. Voici un exemple de l'application d'une étape de 2-opt :



(a) Avant 2-opt **(b) Après 2-opt**

Figure 2.5 Exemple d'application d'une étape de 2-opt.

L'algorithme s'arrête s'il n'a pas pu améliorer la route entre deux itérations, autrement dit, si toutes les tentatives de déconnexions/reconnexions croisées possibles ont mené à une solution ayant un coût supérieur. Introduisons maintenant quelques notions liées à la recherche locale et qui seront illustrées avec le TSP et l'algorithme **2-opt** [32] :

- La fonction objective d'un problème d'optimisation est l'équation décrivant le critère optimisé. Dans le cas du TSP, elle est définie par l'équation : $\mathbf{min}\sum_{e\in E} c_e x_e$, qui indique qu'on doit minimiser le coût total du cycle visitant chaque ville.
- L'espace des solutions est l'ensemble des solutions possibles pour le problème. Pour le TSP, ceci correspond à l'ensemble des cycles différents qui passent par toutes les villes.
- La solution courante est la solution actuelle qui doit être modifiée par l'algorithme pour essayer de trouver une meilleure solution.
- La solution obtenue après une modification de la solution courante est appelée voisin. Dans l'exemple du 2-opt, la figure **2.5 (a)** est la solution courante et la figure **2.5 (b)** est un de ses voisins.
- Le voisinage est l'ensemble des voisins. Pour le 2-opt, il s'agit de l'ensemble des solutions obtenues en effectuant toutes les déconnexions/reconnexions croisées possibles.
- L'opérateur de voisinage est la fonction définissant les modifications à faire à la solution courante pour générer son voisinage. Pour le 2-opt, l'opérateur de voisinage est la déconnexion/-reconnexion croisée.
- Un mouvement est le procédé permettant de générer un voisin de la solution courante. Par exemple, "échange du site **5** et **10**" pourrait être un mouvement dans lequel on inter change la position des sites **5** et **10** dans l'itinéraire d'un véhicule. La figure **2.5** montre l'application d'un mouvement **2-opt**.
- Une recherche locale est un processus itératif où à chaque itération on applique un mouvement à la solution courante. Pour produire la première solution courante qui permettra de démarrer l'algorithme, on utilise souvent une méthode heuristique constructive, car elles sont rapides à exécuter et la solution trouvée a généralement une fonction objective déjà assez bonne.

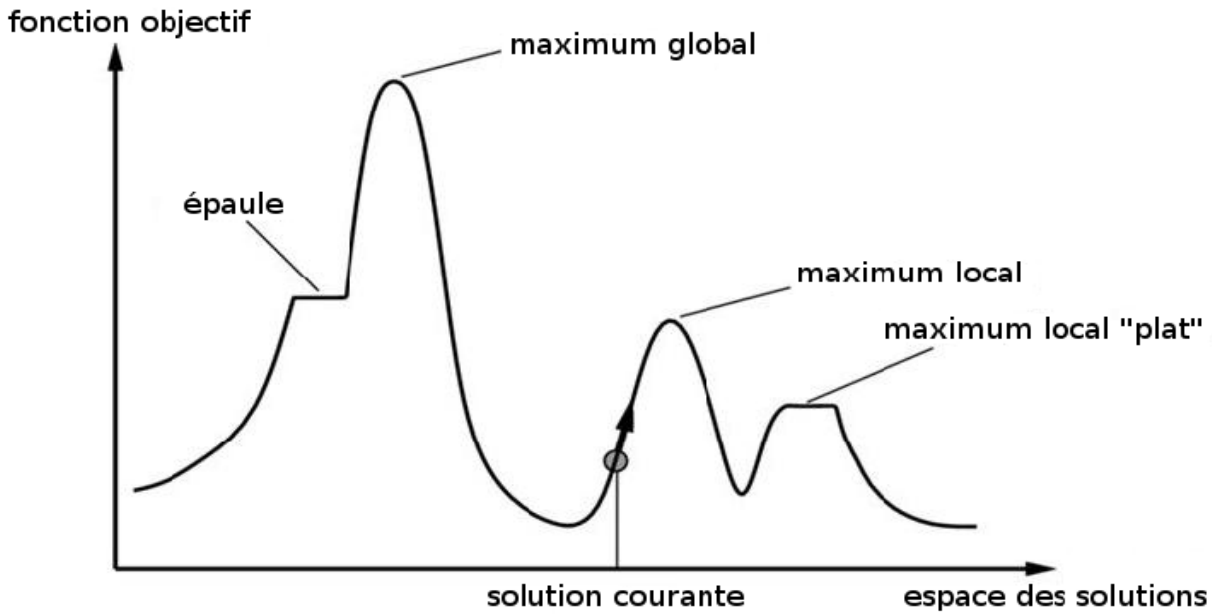


Figure 2.6 Représentation de la fonction objective dans l'espace des solutions [32].

Dans cette figure, on représente en abscisse l'espace des solutions de notre problème. On considère que deux solutions similaires sont proches l'une de l'autre sur cet axe. Autrement dit, les voisins d'une solution sont les points qui sont proches de lui. En ordonnée, on représente la fonction objective du problème. Dans cet exemple, il s'agit d'un problème de maximisation, on cherche à trouver le maximum global, c'est-à-dire la solution pour laquelle la fonction objective est maximale. On parle de maximum local lorsque les points dans le voisinage de ce maximum local ont une valeur de fonction objectif inférieure ou égale.

Établissons une stratégie simple de recherche locale : on choisit parmi les voisins de la solution courante celui dont la fonction objectif est la plus élevée et on répète cela jusqu'à ne plus pouvoir améliorer la solution. On appelle cette stratégie le hill climbing [33] (escalade de colline). En utilisant cette stratégie basique en partant de la solution courante indiquée sur le schéma, on arrive au point correspondant au maximum local.

Ce petit exemple permet de voir les difficultés que cette stratégie basique peut rencontrer. En effet, si la solution courante n'a pas de voisins ayant une meilleure valeur de fonction objective, l'algorithme y est bloqué et s'arrête. Ce problème peut survenir dans le cas de maximums locaux, mais aussi en cas de zones plates comme une épaule et empêchera souvent l'algorithme de trouver le maximum global. Il est donc nécessaire de mettre en place des stratégies plus sophistiquées pour éviter ce genre de problème.

Les algorithmes de recherche locale sont souvent décrits en utilisant les notions d'intensification et de diversification.

- L'intensification est le fait d'explorer une zone prometteuse de l'espace des solutions dans le but d'y trouver une meilleure solution.
- La diversification est le fait d'explorer des zones encore inexplorées de l'espace des solutions pour y trouver éventuellement une meilleure solution.

Avec ces deux nouvelles notions, on voit que notre stratégie de Hill climbing [33] ne fait en réalité que de l'intensification, on cherche en permanence à améliorer la solution. Elle manque de diversification pour pouvoir se débloquer des maximums locaux et des zones plates. Un bon algorithme de recherche locale est donc un algorithme qui arrive à trouver l'équilibre entre intensification et diversification. L'intensification doit permettre d'améliorer la meilleure solution connue, tandis que la diversification doit élargir la zone de recherche. Le fait de pouvoir accepter de dégrader la solution est souvent un élément clé au succès des méthodes de recherche locale, car cela permet d'explorer une plus grande partie de l'espace de recherche et donc de potentiellement trouver le maximum global. Nous allons voir maintenant des algorithmes de recherche locale plus sophistiqués qui sont utilisés pour le VRP :

Recuit simulé : Cette méthode est un algorithme de recherche locale basé sur un processus métallurgique où l'on contrôle le refroidissement d'un matériau pour atteindre son point le plus stable. Cette technique a été adaptée aux problèmes d'optimisation. On démarre avec une haute température que l'on va faire progressivement diminuer, cette température peut être vue comme un paramètre influençant directement la diversification de l'algorithme. Lorsque la température est élevée, la diversification est importante, et inversement lorsqu'elle est basse.
A chaque itération, on choisit aléatoirement un candidat dans le voisinage de la solution courante :

- Si c'est une meilleure solution, elle est acceptée et devient la solution courante (intensification).
- Si c'est une solution moins bonne, elle est acceptée avec une certaine probabilité. Cette probabilité est influencée par deux paramètres :
  - la température actuelle, plus elle est élevée, plus la probabilité augmente
  - la différence de qualité des solutions, si on dégrade beaucoup la solution, la probabilité diminue.

Cette méthode, associée à une recherche à voisinage large, a eu récemment de très bonnes performances pour le problème **CVRP** (VRP avec contrainte de capacité) [34].

Recherche tabou : Ici, l'idée est de choisir à chaque étape le voisin améliorant (ou diminuant le moins) la qualité de la solution courante (intensification). On introduit ensuite un mécanisme d'interdiction de certains mouvements (diversification). Effectivement, sans cette

restriction, le risque de se trouver dans un cycle où l'on visite tout le temps les mêmes solutions est très grand.

Il y a plusieurs stratégies à mettre en place :

- la stratégie d'interdiction : elle décide quels sont les mouvements à interdire
- la stratégie d'autorisation : elle décide quand un mouvement redevient disponible
- la stratégie à court terme : elle permet d'autoriser un mouvement normalement interdit,

Par exemple si ce mouvement mènerait à une amélioration de la meilleure solution connue. Pour voir comment cet algorithme peut être appliqué en pratique pour le VRP. [35].

Cette étape de la solution n'est en soit pas indispensable mais permet généralement d'améliorer les solutions générées pour un faible coût computationnel. Il existe beaucoup d'autres algorithmes utilisables pour cette étape, puisqu'on peut utiliser tous ceux utilisés pour résoudre le TSP.
On notera notamment l'heuristique de Lin-Kernighan [36], qui est parfois utilisée pour ce type de solution. Elle est particulièrement adaptée pour des problèmes où le nombre de sites desservis par véhicule est élevé. Elle ne sera pas utilisée ici car son implémentation est plus complexe et nous utiliserons des instances de taille relativement modeste lors des expériences.

2.4.1.5 Mise à jour des phéromones

La dernière étape de la solution est de mettre à jour les pistes de phéromones. Il existe de nombreuses manières différentes de procéder, c'est généralement cette étape qui différencie le plus les différents solutions utilisant l'optimisation par colonie de fourmis.

Le but ici est donc de modifier les quantités de phéromones présentes sur les arêtes afin de mieux guider les prochaines fourmis qui construiront leur route. Dans un premier temps, on les fait s'évaporer car, comme dans la réalité, un chemin qui n'est plus emprunté perd au fur et à mesure sa piste de phéromones. Ensuite, on augmente leur quantité sur les arêtes appartenant à la solution construite. On distingue deux contributions à la quantité de phéromones que l'on ajoute sur une arête de la solution :

- si la solution est de bonne qualité, on en ajoute plus
- si l'arête participe à la qualité de la route à laquelle elle appartient, on en ajoute plus.

Mathématiquement, voici la formule utilisée pour mettre à jour les phéromones sur chaque arête :

$$T_{ij}^{new} = \begin{cases} p.\ T_{ij}^{old} + \Delta T_{ij} & \text{si l'arête } \textbf{(i, j)} \text{ est dans la solution} \\ p.\ T_{ij}^{old} & \text{sinon.} \end{cases} \quad \textbf{(2.3)}$$

p est un paramètre à fixer pour simuler l'évaporation, il doit être compris entre **0** et **1**. Une valeur de **0,9** correspond par exemple à une évaporation de **10%** de la quantité déjà présente.

ΔT_{ij} est l'incrément de la quantité de phéromones pour l'arête ***(i, j),*** il est défini comme ceci :

$$\Delta T_{ij} = \frac{Q.K}{L} \cdot \frac{D - c_{ij}}{m.D} \quad \textbf{(2.4)}$$

- **Q** est un paramètre permettant de fixer l'échelle
- **K** est le nombre de véhicules utilisés
- **L** est le coût total de la solution
- **D** est le coût total de la route à laquelle l'arête appartient
- **m** est le nombre de sites desservis sur la route
- **cij** est le coût entre **i** et **j**

On voit dans cette formule que **Q·K**

L est la contribution due à la qualité globale de la solution.

Tandis que **D−cij**, **m·D** est la contribution liée à la qualité de la route de l'arête.

Finalement, on fixe une borne minimale et maximale aux quantités qui peuvent se trouver sur une arête.

La borne minimale sert à éviter qu'une arête ne puisse plus jamais être sélectionnée par une fourmi. En effet, si la quantité de phéromones vaut **0**, d'après la formule probabiliste **2.2** montrée à l'étape de construction des routes, la probabilité de sélectionner l'arête est de **0**.

La borne maximale sert à éviter qu'une arête soit tout le temps sélectionnée. Même si la sélection de l'arête mène souvent à de bonnes solutions, elle risque d'empêcher la découverte d'autres routes pouvant être de meilleure qualité.

Ces bornes sont fixées comme ceci :

$$T_{min} = \frac{Q}{\sum_i 2c_{0i}} \text{ et } T_{max} = \frac{Q}{\sum_i c_{0i}} \quad \textbf{(2.5)}$$

- **Q** est la même constante que celle définie ci-dessus
- $\mathbf{c_{0i}}$ est le coût entre le dépôt et le site **i**

Ces bornes ont été déterminées de manière empirique par [37] et ont été reprises pour cette implémentation, elles n'ont pas de signification particulière. En effet, on remarque que ce qui importe dans la formule probabiliste **2.2**, ce sont les quantités relatives de phéromones et non les valeurs en elles-mêmes, il est donc cohérent de les fixer "arbitrairement".

2.4.1.6 Redémarrages

Après un grand nombre d'itérations de cet algorithme, il est intéressant de faire un redémarrage. En effet, certaines arêtes étant fortement marquées par les phéromones sont empruntées presque systématiquement car la probabilité qu'elles soient sélectionnées est très élevée. Les solutions trouvées sont donc souvent similaires et ne permettent plus d'améliorer la meilleure solution connue.

Un redémarrage consiste à réinitialiser toutes les valeurs de phéromones associées aux arêtes afin de permettre aux fourmis de découvrir, éventuellement, de nouvelles routes.

2.4.2 Optimisation linéaire en nombres entiers

L'optimisation linéaire en nombres entiers est une méthode que l'on peut appliquer uniquement aux problèmes pouvant être décrits par un modèle mathématique ayant certaines propriétés :

- les variables utilisées doivent prendre des valeurs positives entières
- les contraintes imposées sur ces variables doivent être linéaires
- la fonction objective doit également être linéaire.

Un problème **ILP** peut être écrit sous sa forme dite canonique :

$$\textbf{Min } \mathbf{c^T x} \qquad \textbf{(2.6)}$$

Tel que

$$\mathbf{Ax \leq b} \qquad \textbf{(2.7)}$$

$$\mathbf{x \geq 0} \qquad \textbf{(2.8)}$$

$$\mathbf{x \in Z^n} \qquad \textbf{(2.9)}$$

Notons **n** le nombre de variables et **m** le nombre de contraintes. **x** est un vecteur de dimension **n** contenant les variables. Les vecteurs colonne **b** de taille **m** et **c** de taille **n**, et la matrice **A** de taille **m × n** contiennent des coefficients à valeurs entières. L'équation **2.6** indique que la fonction objective est une combinaison linéaire des variables. Il s'agit ici d'un problème de minimisation, mais la définition de la forme canonique pour les problèmes de maximisation est bien sûr identique. L'équation **2.7** est l'équation indiquant que les contraintes sur les variables doivent être linéaires. Les équations **2.8** et **2.9** imposent que les variables soient positives et entières. On appelle généralement la contrainte **2.9**, la contrainte d'intégralité.

On remarque que la formulation mathématique du VRP présentée à la section **1.2** peut être facilement mise sous cette forme canonique, l'optimisation linéaire en nombres entiers peut donc être utilisée pour le résoudre.

Voyons un exemple de problème **ILP** sous forme canonique :

$$\textbf{Max } x2$$

$$-x1 + x2 \leq 1$$

$$3x1 + 2x2 \leq 12$$

$$2x1 + 3x2 \leq 12$$

$$x1, x2 \geq 0$$

$$x1, x2 \in \mathbf{Z}$$

Pour cet exemple, **x**, **c**, **b** et **A** sont donnés par :

$$\mathbf{X} = \begin{bmatrix} x1 \\ x2 \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 12 \\ 12 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} -1 & 1 \\ 3 & 2 \\ 2 & 3 \end{bmatrix}$$

On peut représenter graphiquement cet exemple dans un espace à deux dimensions où les axes correspondent aux variables ***x*1** et ***x*2** :

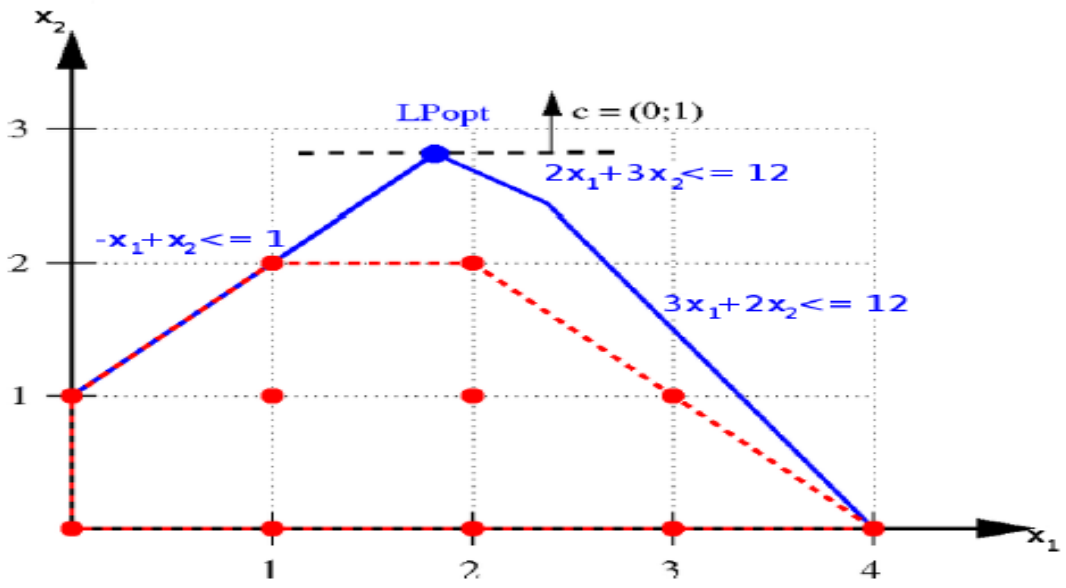


Figure 2.7 Représentation graphique de l'exemple de problème ILP décrit ci-dessus [38].

Les contraintes sont représentées en bleu. La ligne en pointillé noir représente la fonction objective, la flèche vers le haut indique qu'il s'agit d'une maximisation. Les points rouges sont les points dits réalisables, ce sont les points qui respectent toutes les contraintes, y compris celle d'intégralité. Les lignes en pointillés rouges représentent l'enveloppe complexe, il s'agit du plus

petit polyèdre contenant tous les points réalisables. Pour mieux visualiser ce que représente cette enveloppe complexe, il peut être utile de faire une analogie avec un élastique entourant un ensemble de points:

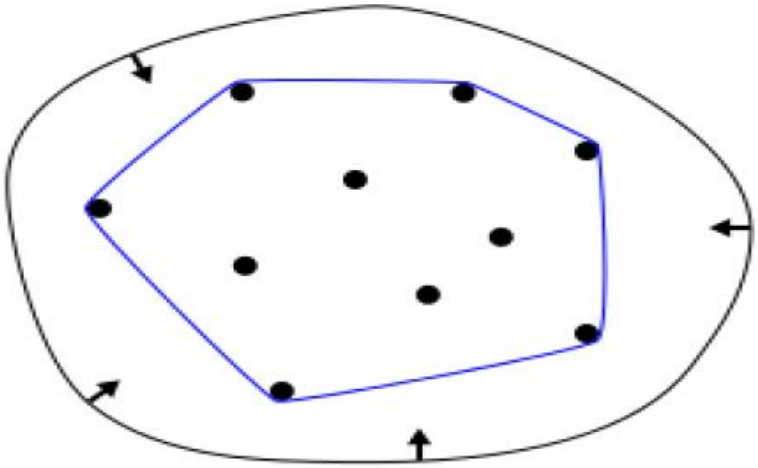


Figure 2.8 Analogie entre un élastique entourant un ensemble de points et leur enveloppe complexe [39].

Voyons maintenant le fonctionnement de l'algorithme qui est utilisé dans la solution :

2.4.2.1 Algorithme d'optimisation linéaire en nombres entiers

L'algorithme présenté ici est basé sur le principe de séparation et évaluation (branch and bound, en anglais). Ce principe est formulé en deux étapes qui sont présentées ci-dessous. L'étape de séparation (ou branchement) consiste à partitionner un problème **P** en sous problèmes **Pi** tel que le coût de la solution trouvée en résolvant **P** est le même que celui de la meilleure solution trouvée en résolvant chacun des sous-problèmes **Pi**. Par exemple, pour le problème ILP ci-dessous, une procédure de séparation valide est de partitionner *P* en sous-ensembles **P1, ..., Pk** :

$$\min c^T \mathbf{x}$$

$$\mathbf{x} \in p$$

est équivalent à résoudre les sous-problèmes pour $i \in 1, ..., k$ et garder la meilleur solution trouvée :

$$\min c^T \mathbf{x}$$

$$\mathbf{x} \in p_i$$

On peut ensuite continuer à appliquer ce principe de séparation aux sous-problèmes, on se retrouve alors avec une structure en arbre comme ceci :

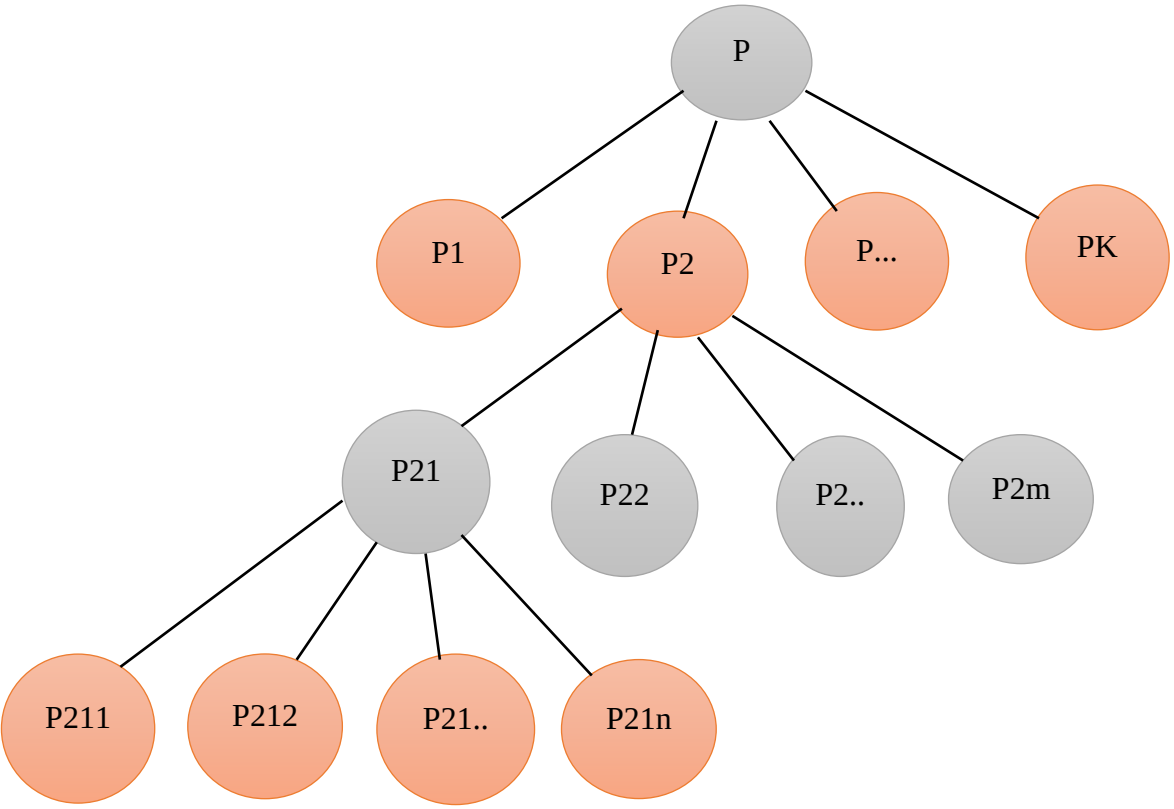


Figure 2.9 Structure en arbre formée lors des séparations successives des problèmes en sous-problèmes.

- On appelle communément cette structure l'arbre de recherche.
- L'étape d'évaluation consiste à calculer une borne sur la fonction objectif pour un sous problème.
- Le but de cette borne est d'essayer de déterminer si la meilleure solution connue actuellement peut être améliorée à partir de ce sous-problème. Si la borne obtenue indique que ce n'est pas possible, alors il n'est pas nécessaire de le résoudre.

Pour le problème présenté ci-dessus, cette borne se traduit par :

$$\mathbf{b(P_i)} \leq \mathbf{min}_{\mathbf{x} \in \mathbf{p_i}} \ \mathbf{c^T x} .$$

b(Pi) est la procédure permettant de générer une borne pour un sous-problème **Pi**. On cherche à ce que cette procédure ait les deux caractéristiques suivantes :

- elle doit être rapide à calculer
- elle doit donner une borne inférieure proche de l'optimum pour le sous-problème.

Dans le cas de problème **ILP,** on utilise couramment la relaxation linéaire, cela consiste à résoudre le problème en retirant la contrainte d'intégralité sur les variables. Si une variable *x* a pour domaine **{0, 1},** sa version relaxée aura l'intervalle **[0, 1]** pour domaine. Effectivement, en retirant

la contrainte d'intégralité, cela revient à résoudre un problème d'optimisation linéaire classique, pour lequel des algorithmes efficaces sont connus. L'un d'entre eux est l'algorithme du simplexe. L'algorithme du simplexe se base sur la notion de polytope convexe. Un polytope convexe est un objet géométrique défini dans n'importe quel nombre de dimensions, ayant des faces planes polygonales et dont l'ensemble de points qui le constitue forment un ensemble de points convexe. L'ensemble des contraintes d'un problème d'optimisation linéaire forme un polytope convexe. Sur la figure **2.7**, le polytope convexe formé est celui défini par les contraintes, en bleu, et les axes **x** et *y*. L'enveloppe complexe, en pointillé rouge est également un polytope convexe. L'algorithme du simplexe se base sur une propriété très importante du polytope convexe formé par les contraintes : au moins un de ses sommets est nécessairement une solution optimale du problème. Cette propriété a également un intérêt particulier pour l'enveloppe complexe. Puisqu'il s'agit d'un polytope convexe et que tous ses sommets sont des points réalisables, une des solutions optimales d'un problème ILP est un des sommets de son enveloppe complexe.

L'algorithme du simplexe est un processus itératif démarrant en l'un des sommets du polytope convexe formé par les contraintes linéaires.

A chaque itération, on se déplace le long de l'arête menant à la plus grande amélioration de la fonction objective. Lorsqu'il n'est plus possible de trouver un sommet améliorant la fonction objectif, l'algorithme s'arrête et le sommet courant est la solution optimale.

Passons maintenant au solveur pour le VRP en lui-même, voici un schéma montrant l'architecture globale :

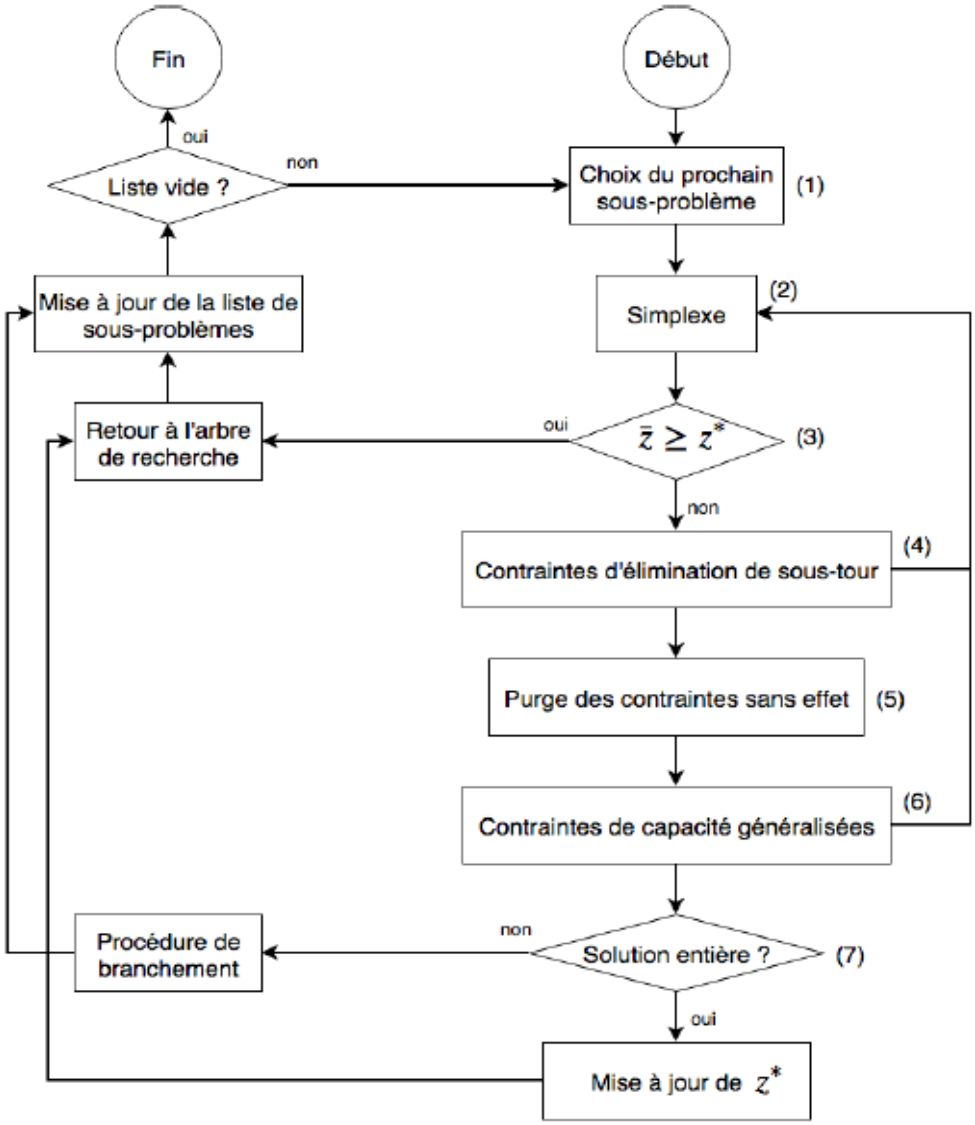


Figure 2.10 Schéma global de la solution [40].

On retrouve dans ce schéma les notions et algorithmes expliqués précédemment, on va passer en revue les différentes étapes.

Étape (1). On commence par choisir le prochain sous-problème à traiter, par rapport à notre analogie avec l'arbre de recherche, cela correspond donc à un nœud de l'arbre.

Étape (2). On utilise la relaxation linéaire sur ce nœud pour trouver une borne, on utilise donc l'algorithme du simplexe pour résoudre le problème d'optimisation linéaire. Les contraintes utilisées sont donc celles présentées dans le modèle mathématique du VRP, section **2.2,** mais sans les contraintes d'intégrité.

Étape (3). On compare la borne obtenue, notée $\bar{\mathbf{z}}$ avec la meilleure solution connue actuellement, notée $\mathbf{z}^*$ Si la borne est moins bonne que $\mathbf{z}^*$, il n'est pas nécessaire de continuer à résoudre pour ce nœud et on retourne à l'arbre de recherche.

Étape (4). On vérifie ensuite que la solution ne contient pas de sous-tour. Par sous-tour, on entend un cycle de sites ne contenant pas le dépôt et donnant donc lieu à une solution invalide.
Il est nécessaire de vérifier cela car les contraintes d'élimination de sous-tour sont introduites de manière lazy, c'est-à-dire qu'elles sont introduites seulement lorsque c'est nécessaire. En effet, l'équation**1.13** empêchant la sous-tour définie dans le modèle mathématique du VRP donne lieu en pratique à un nombre exponentiel de contraintes. Il serait donc inapproprié d'essayer de tenir compte d'autant de contraintes dans le modèle de base.
Étape (5). La purge des contraintes sans effet consiste à retirer du modèle les contraintes n'ayant plus d'utilité. Ce cas peut se présenter par exemple avec les contraintes de sous-tour qui sont ajoutées au cours de l'exécution, si certaines d'entre elles ne sont plus violées depuis un grand nombre d'itérations, elles peuvent être retirées du modèle. Elles seront à nouveau ajoutées plus tard si nécessaire. Cela permet d'alléger le modèle et donc d'accélérer la recherche.
Étape (6). On vérifie maintenant que la solution respecte les contraintes de capacité. Pour la même raison qu'à l'étape **4**, on rajoute ces contraintes de manière lazy pour ne pas surcharger, le modèle.
Étape (7). Finalement, on regarde si la solution est entière ou non. Une solution entière est une solution dont les variables prennent toutes des valeurs entières et qui respecte donc les contraintes d'intégrité. Cette étape est effectivement nécessaire puisqu'on a utilisé la relaxation linéaire à la deuxième étape. Si elle est entière, alors on met à jour la meilleure solution connue si la solution trouvée est meilleure, sinon, on applique la procédure de branchement et on rajoute les nouveaux sous-problèmes à la liste de sous-problèmes.

2.4.2.3 Procédure de branchement

Supposons que la solution optimale du problème linéairement relaxé, noté **P**, contienne une variable $x1$ de valeur **2.7**. On peut alors séparer le problème en deux sous-problèmes **P1** et **P2**, dans **P1** on ajoutera la contrainte $\mathbf{x \leq 5}$ et dans **P2**, on ajoutera la contrainte $\boldsymbol{x \geq 6}$. Comme expliqué précédemment, la meilleure solution trouvée en résolvant
P1 et **P2** sera la même que celle trouvée en résolvant directement **P**.
Dans le modèle utilisé à l'étape **6**, des contraintes ont été ajoutées afin d'améliorer les performances. Ces contraintes, appelées user cuts, sont ajoutées uniquement dans le but d'accélérer la recherche et ne sont pas requises pour garantir l'optimalité de la solution trouvée. Elles, permettent de réduire la taille de l'espace de recherche sans en retirer de solution réalisable.
Celles qui ont été utilisées ici sont détaillées dans l'étape 5 de la section **3** de [40] (ceci n'est pas expliqué ici, car cette partie de la solution sert uniquement à améliorer ses performances).

L'algorithme qui a été présenté est une version assez générique de ce qui est utilisé dans les solutions d'optimisation linéaire.

2.5. Approche choisie

Comme expliqué précédemment, il existe deux grands types d'approches pour résoudre le VRP : les méthodes exactes et les méthodes approchées. Les méthodes exactes ont l'avantage de donner la solution optimale du problème, mais au prix d'un temps d'exécution souvent très long pour les instances de grande taille. Les méthodes approchées permettent de résoudre ces instances de grande taille, mais la solution trouvée n'est pas nécessairement la solution optimale.
L'idée de réunir ces deux types de méthodes semble donc très intéressante pour créer des algorithmes plus performants, les auteurs proposent une approche permettant la collaboration de solutions de problème de satisfaction de contraintes (CSP, Constraint Satisfaction Problem) et de la version avec optimisation de fonction objectif (CSOP, Constraint Satisfaction Optimization Problem). Leur approche utilise deux types de solutions : un solveur aidé, noté i-solver et le solveur aidant, noté c-solver.

Une des techniques proposées est d'utiliser un solveur exact comme i-solver et un solveur approché comme c-solver. Lorsqu'i solveur se trouvent bloqué, ne parvenant plus à trouver de meilleures solutions, il passe le contrôle à c-solver, qui utilisera une méthode approchée, telle que de la recherche locale pour trouver rapidement une nouvelle solution. Le but est que l'aide apportée par c-solver permette de réduire la taille de l'arbre de recherche en se focalisant sur les branches ayant une plus grande probabilité de contenir une solution optimale.

Une autre méthode combinant méthode exacte et approchée est proposée : la recherche locale est utilisée pour accélérer un algorithme branch and bound au prix de la garantie d'optimalité. Au moment de la publication, cette méthode avait les meilleures performances pour certains CSP tels que le n-queens problem, Il existe également d'autres approches hybrides qui ont été utilisées pour résoudre le VRP :

- utilise l'optimisation par colonie de fourmis et un algorithme génétique
- combine deux algorithmes de recherche locale : la recherche tabou et le recuit simulé
- utilise l'optimisation par colonie de fourmis et le recuit simulé pour résoudre le VRP avec dépôts multiples et contrainte de fenêtre de temps
- utilise une succession d'heuristiques constructives et perturbatrices.

- L'approche qui a été choisie dans ce travail est une collaboration entre une méthode exacte, l'optimisation linéaire en nombres entiers et une méthode approchée, l'optimisation par colonie de fourmis.

- Le rôle de la solution ACO sera de fournir des solutions de qualité acceptable rapidement à la solution ILP.
- Le rôle de la solution ILP sera d'essayer de trouver la solution optimale du problème avec l'aide des solutions trouvées par la solution ACO. Comme expliqué dans la section **2.6.2.1**, un des points clés de l'algorithme branch and bound est l'étape d'évaluation. C'est dans cette partie que l'on pourra utiliser les solutions trouvées par la solution ACO. Le coût de ces solutions pourra être utilisé comme borne de la fonction objective, et lorsqu'un sous-problème ne pourra pas conduire à une solution de coût inférieur à cette borne, il ne sera pas nécessaire de le résoudre. Ceci devrait avoir pour effet de réduire la taille de l'arbre de recherche et donc d'accélérer le temps de résolution des instances du VRP.

Conclusion

Dans ce chapitre, nous avons essayé d'aborder quelques notions de base en optimisation. Nous avons présenté des définitions de notions confrontées souvent dans le domaine de l'optimisation en allant de la définition d'un problème d'optimisation de tournées de véhicules à capacité limitée, passant par les types des problèmes d'optimisation jusqu'à la définition d'une solution optimale d'un problème donné. Ensuite, nous avons dévoilé les processus d'optimisation et Les méthodes de résolution de problèmes d'optimisation (colonie de fourmis et linéaire en nombres entiers).

Dans le chapitre qui suit, nous présentons la modélisation de problème d'optimisation de tournée de véhicule avec capacité limitée.

Chapitre 03

MODILISATION DE PROBLEME DES TOURNEES DES VEHICULES HETEROGENES

Introduction

Ce chapitre est consacré à la conception de la structure et du fonctionnement du système proposé. À cet effet, nous définissons tout d'abord les besoins de site en termes d'exigence fonctionnelles et non fonctionnelles. Puis, nous décrivons une vue approfondie qui explique les diagrammes des classes en abordant la partie conception détaillée. Enfin, nous développons quelques scénarios en se basant sur les diagrammes de séquences qui représentent une vue dynamique de notre site.

3.1. Spécification des besoins

Dans cette partie nous étudions les besoins fonctionnels et non fonctionnels de notre système.

3.1.1. Besoins fonctionnels

Les besoins fonctionnels constituent une sorte de promesse ou de contrat au comportement d'application générée. Donc, dans cette partie nous représentons une description du diagramme des cas d'utilisation.

3.1.1.1. Diagramme de cas d'utilisation

Ce diagramme permet de formaliser les besoins. C'est le diagramme principal du modèle UML [41] celui où s'assener la relation entre l'utilisateur et les objets que le système met en œuvre.

Dans notre projet nous avons localisé un principal qui est l'utilisateur ; un utilisateur peut gérer un problème c .à. d qu'il peut l'ajouter ou le supprimer. Il peut aussi résoudre un problème en choisissant son type et l'instance qu'il va la résoudre.

Une autre fonctionnalité consiste dans la visualisions de la solution générée.

Il peut aussi gérer une instance du problème soit par l'ajout, soit par la suppression, Soit par la modification.

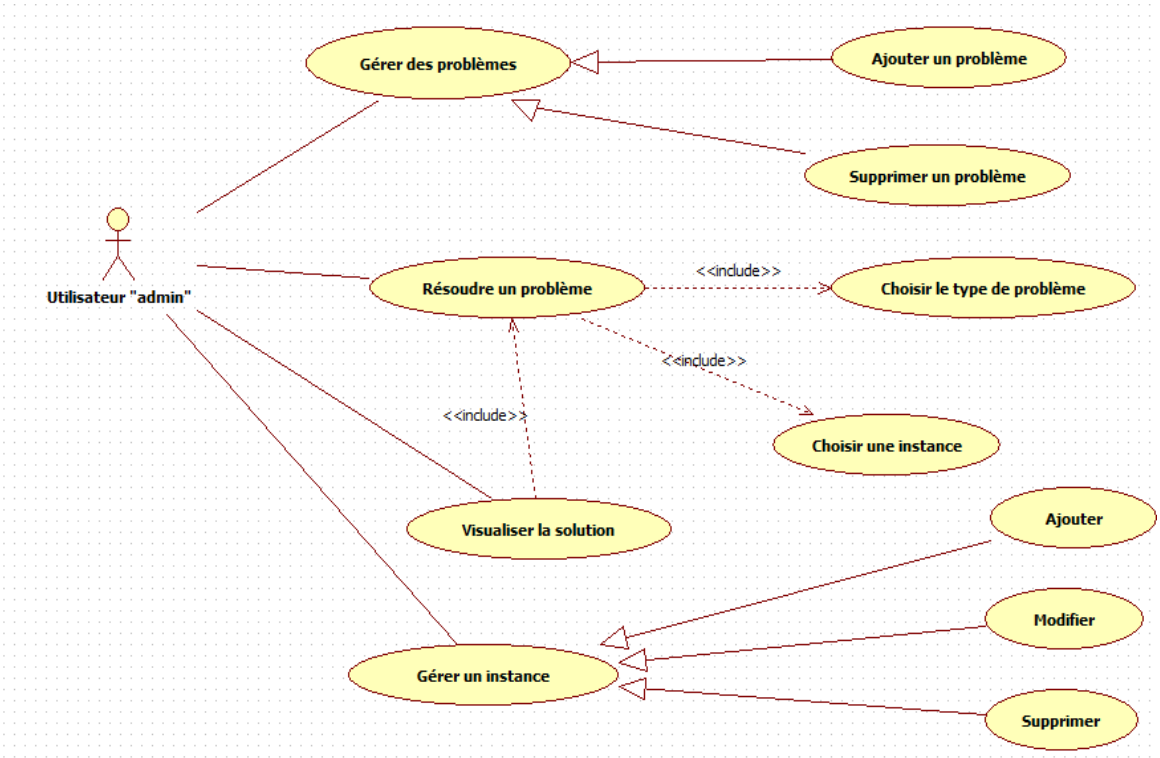


Figure 3.1 Diagramme de cas d'utilisation.

3.1.2. Besoins non fonctionnels

Les besoins non fonctionnels peuvent être considérés comme des besoins fonctionnels spéciaux. Parfois, ils ne sont pas rattachés à un cas d'utilisation particulier, mais ils caractérisent tout le système (l'architecture, la sécurité, le temps de réponse, etc.). Le système doit garantir les besoins opérationnels suivants :

- Le temps de réponse de l'application doit être rapide.
- Les interfaces doivent être conviviales et claires.
- Le système doit être souple pour une extension future (ajouter des nouvelles fonctionnalités
- L'application web doit préserver une bonne qualité en termes de gestion d'erreur.

3.2. Conception

L'approche objet est centrée sur le diagramme de classes qui décrit aussi bien des actions que des informations dans une même entité. Les autres diagrammes nous aident à voir clair dans les besoins et dans la solution qui est à développer. Ils permettent de compléter le diagramme de classes.

3.2.1. Structure statique du système

Elle est représentée sous forme d'un diagramme de classe car c'est le plus utilisé et le plus indispensable. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens. Ces derniers représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation).

3.2.2. Description des classes

Dans notre projet, on a détecté 9 classes dont quatre représentent les problèmes traités dans notre projet. Ces classes sont :

- Classe Dépôt : qui sont caractérisé par un numéro et des coordonnées XD et YD.
- Classe Site : qui sont caractérisé par un numéro et des coordonnées x et y.
- Classe SiteQuantite : qui héritent de " Site ".Mais elle est caractérisée de plus. Par une quantité.
- Classe SiteTw : qui hérite de " SiteQuantite ", mais elle est caractérisée de plus par une fenêtre de capacité.
- Classe PVR: c'est une classe qui représente le problème de voyageur de ramassage.
- Classe PTV : c'est une classe qui représente le problème de tournées des véhicules, elle hérite de PVR et elle est liée à un nombre fini des véhicules qui ont une capacité infinie.
- Classe PTVC : c'est une classe qui représente le problème de tournées des véhicules à capacités, elle hérite de PTV et elle est liée à un nombre fini des véhicules, mais qu'ils ont une capacité finie.
- Classe PTVFT : c'est une classe qui représente le problème de tournées des véhicules à capacité limitée avec ramassage, elle hérite de PTVC et elle est caractérisée par une contrainte de temps qu'il faut la respecter.
- Classe véhicule : qui est caractérisée par une capacité finie ou infinie.

3.2.3. Diagramme de classes

La figure **3.2** représente le diagramme de classes de la solution proposée. Nous avons une classe PVR qui constitue un agrégat pour les deux classes « Site » et « Dépôt ». Elle a aussi un lien avec la classe « Véhicule » et elle à des méthodes telles que la recherche Tabou, chercherChemin, etc.

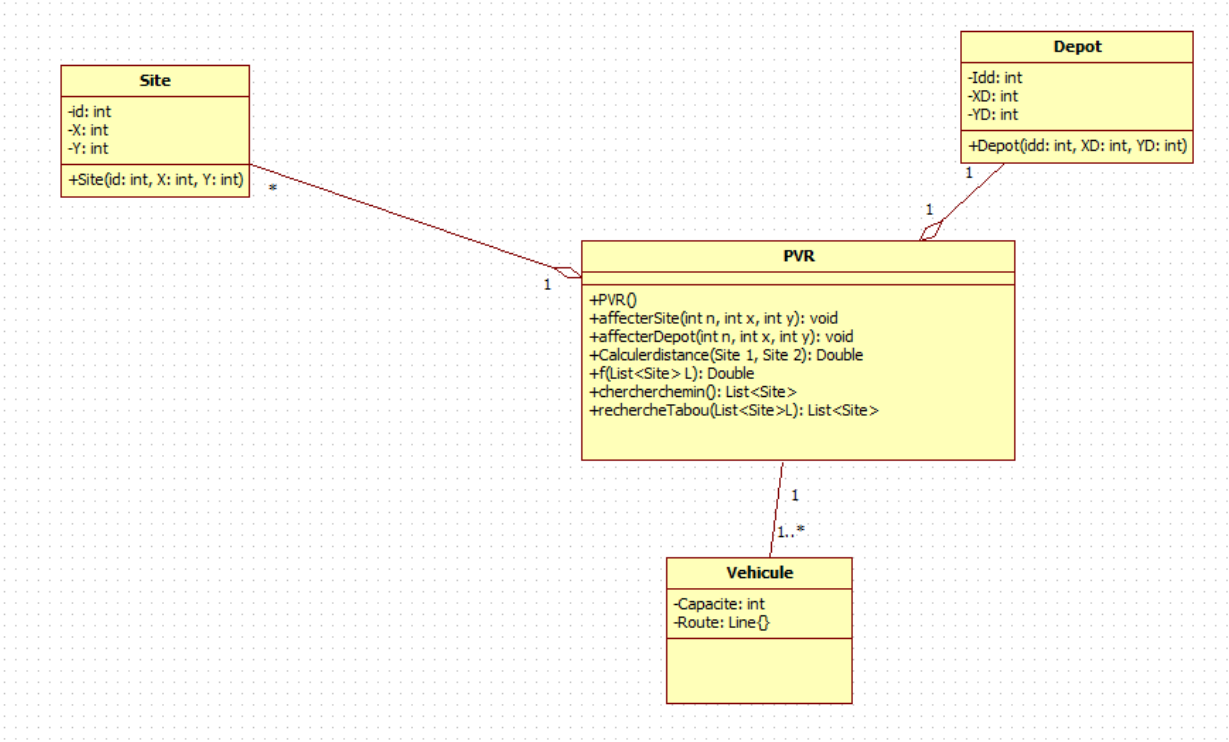


Figure 3.2 Diagramme de classes pour le PVR

Dans la figure **3.3** on a la même présentation que celle de PVC sauf que le nombre des véhicules dans le PTV passe de 1 à plusieurs.

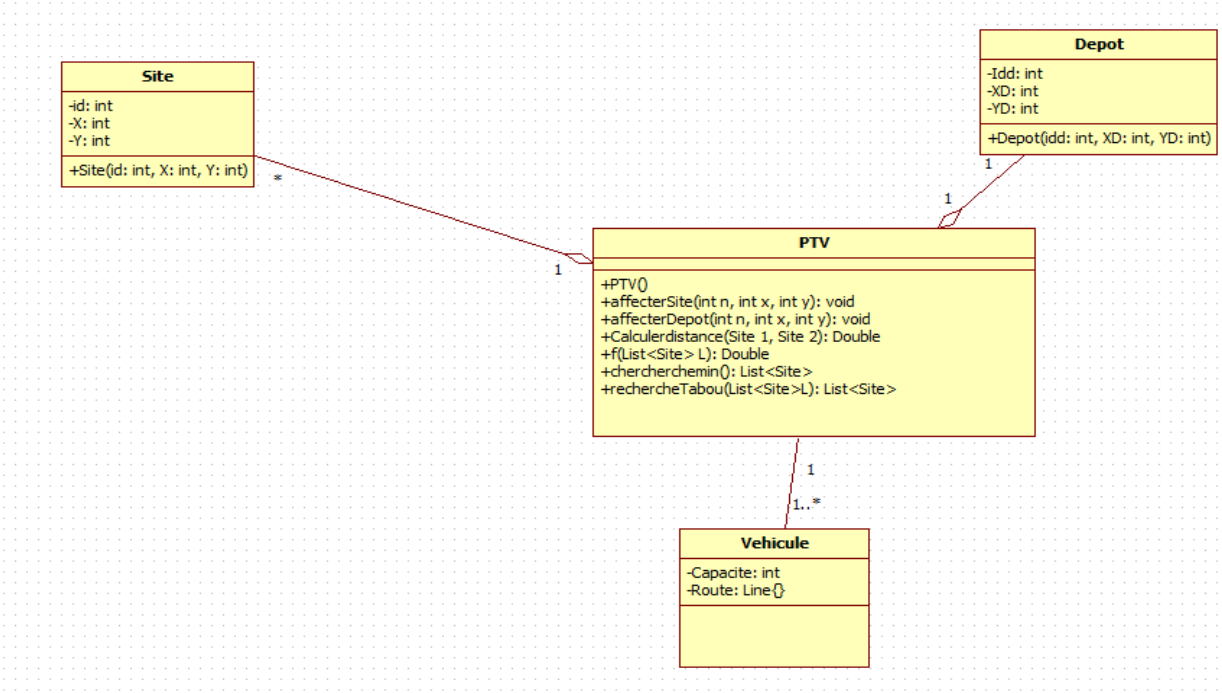


Figure 3.3 Diagramme de classes pour le PTV

Dans la figure **3.4** on a aussi le même principe que les précédentes sauf que pour PTV à capacités le site possède une quantité et l'attribut " capacité " de la classe véhicule va avoir une valeur finie.

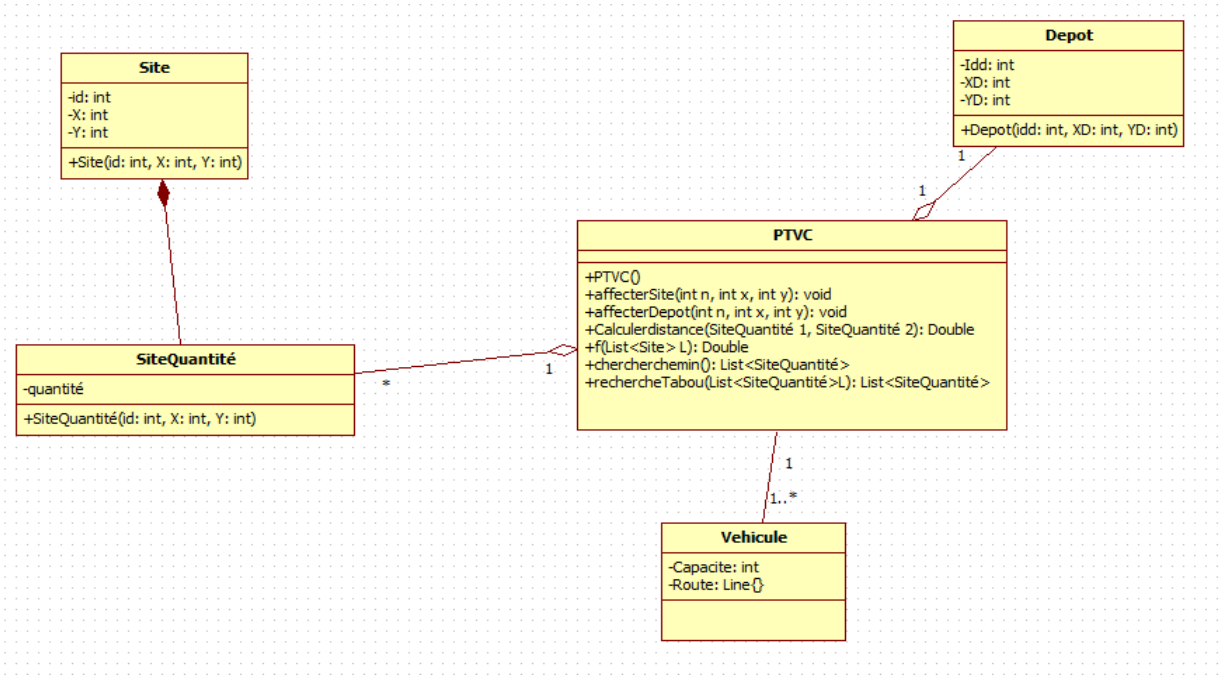


Figure 3.4 Diagramme de classes pour le PTV à capacités

Dans la figure **3.5** on a une présentation du PTV à capacité limitée dans laquelle une classe PTVFT est liée à la classe Véhicules, Dépôt et Site Tw (qui possède à part la quantité, une fenêtre de capacité).

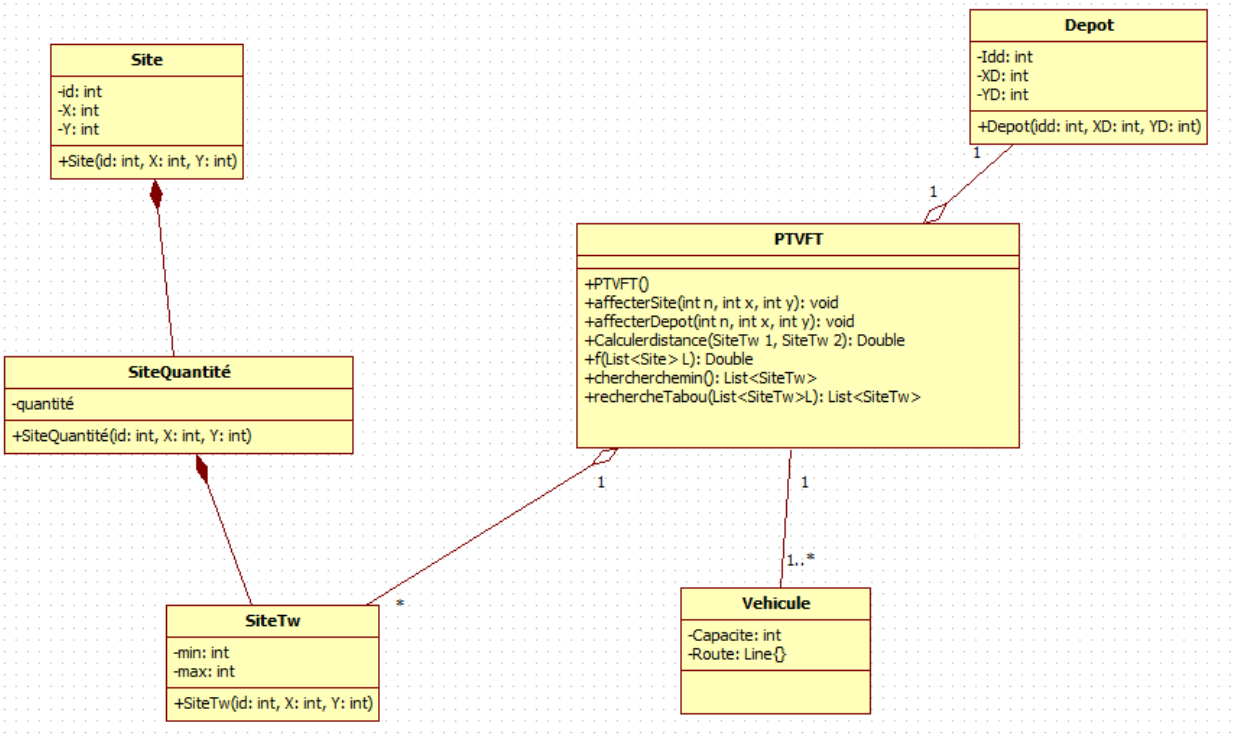


Figure 3.5 Diagramme de casses pour le PTV à fenêtres de capacité.

3.2.3. Diagramme de classes général

Après qu'on a détaillé les diagrammes de classes chacun à part, on va les rassembler ensemble dans un seul diagramme, en cherchant des propriétés communes entre eux.

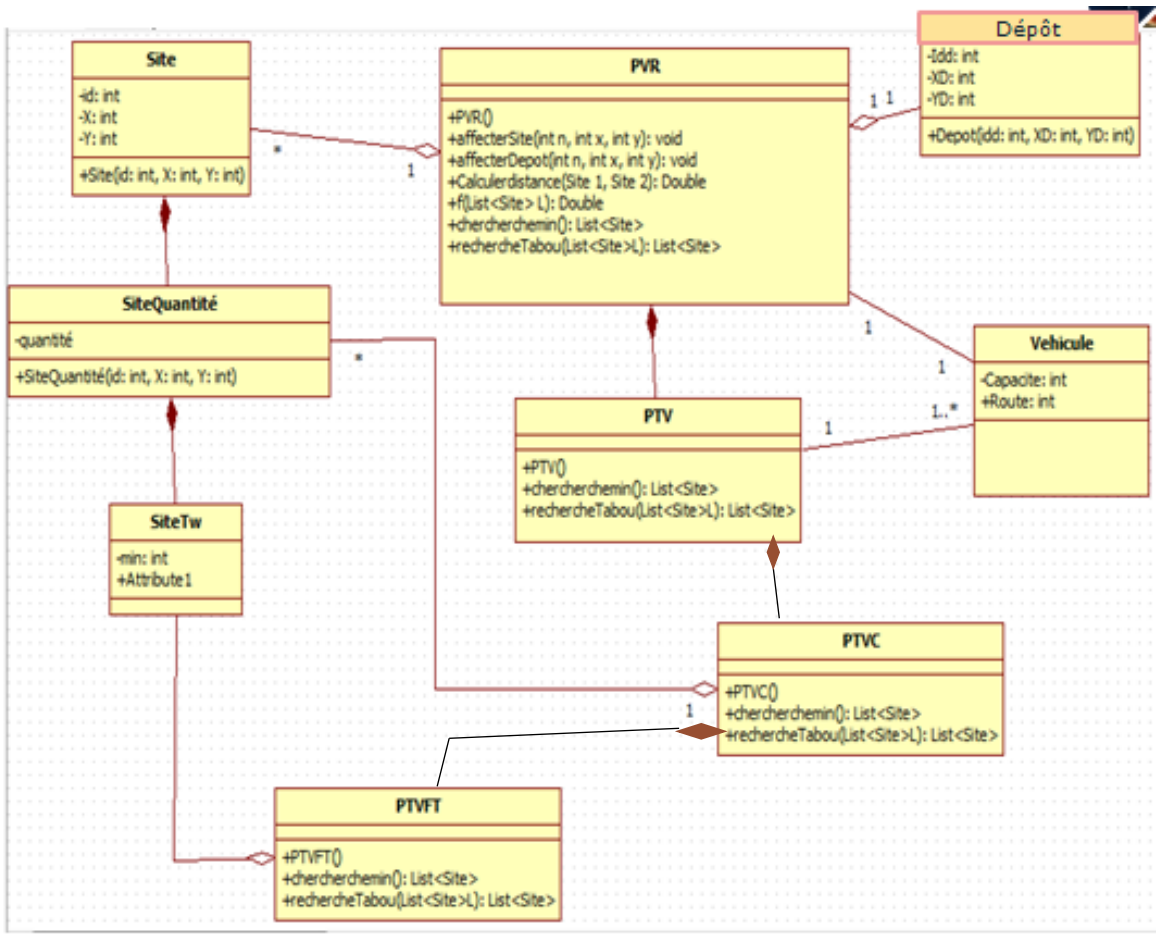


Figure 3.6 Diagramme de classe générale

3.3. Etude dynamique du système

Un diagramme de séquence montre les interactions entre les systèmes arrangés en séquences dans le temps et les messages échangés. Dans cette partie, nous présenterons trois diagrammes de séquences illustrant deux scénarios de notre application.

3.3.1. Diagramme de séquence pour la visualisation d'une solution au problème

La figure **3.7** montre la visualisation d'une solution d'un problème TSP. En effet, lorsque l'utilisateur clique sur un problème Tsp existant dans l'arbre des problèmes, une méthode sélectionner Tsp() de la classe « choix » est invoqué. Dans le code de cette méthode, nous faisons appel à la méthode chargerFichier de la classe Tsp qui a le rôle de chargement du fichier XML ; Ensuite, elle invoque la méthode dessinerRoute de la classe Tsp pour dessiner les routes. Enfin, elle invoque la méthode dessinerVille pour dessiner les villes.

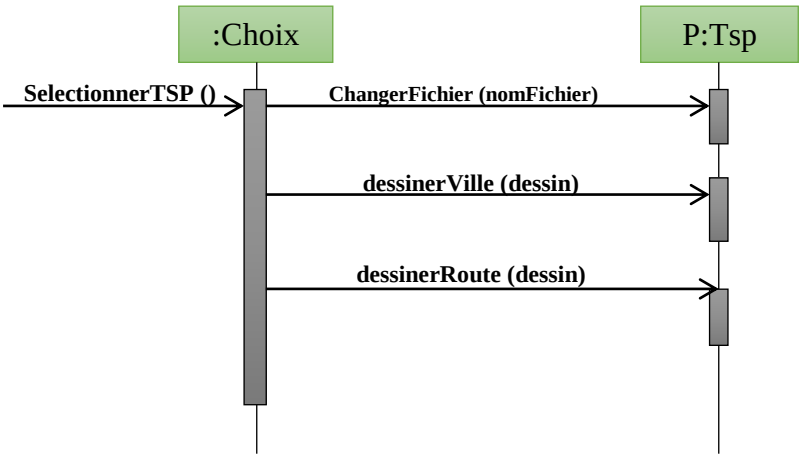


Figure 3.7 Diagramme de séquence de visualisation d'une solution au problème

3.3.2. Diagramme de séquence pour l'ajout d'un problème

Ajout d'un problème VRP Dans ce diagramme de séquence, une méthode " ajouter " du classe " choix " est invoqué, dans laquelle on fait appel aux méthodes :

- SaisirNbVehicule : dans laquelle on fait appel à "SetNbVehicule" de la classe Vrp.
- SaisirDonnées : cette méthode est invoquée n fois jusqu'à satisfaire la condition d'arrêt. Dans cette méthode on test si le compteur ==-1, on appel "affecterDepot" de la classe Vrp. sinon on appelle " affecterSite ". Aussi il y'a appel à la méthode " ajouterLigne " de la classe Vrp. Dans le cas où le bouton supprimer est cliquer une méthode " suppDonnées " de 11 la classe choix est invoquée. Et dans le cas où le bouton enregistrer et cliquer. Une méthode " EnregDonnées " est invoquée dans laquelle on appelle la méthode enregistrer " de la classe Vrp.

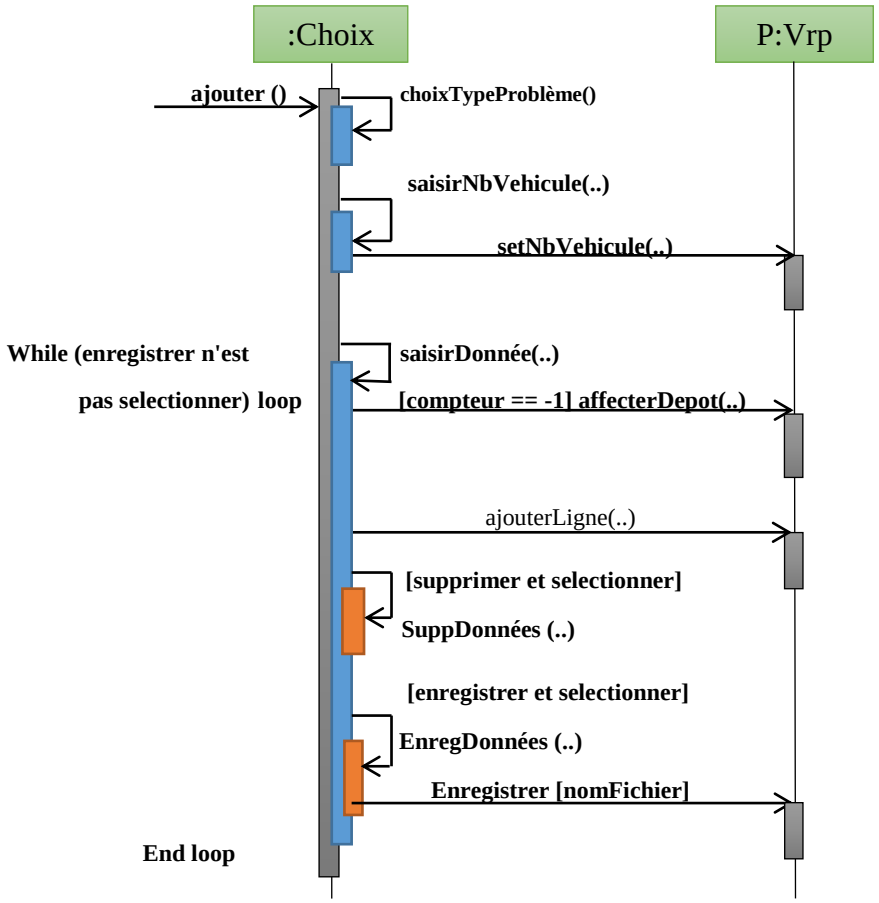


Figure 3.8 Diagramme de séquence pour l'ajout d'un problème

3.3.3. Diagramme de séquence d'une résolution d'instance du VRP avec la collaboration entre solutions

Ce diagramme de séquence montre un exemple simplifié de résolution d'instance du VRP avec le protocole de communication (chapitre 2). Dans cette exécution, la première solution envoyée par un des solutions, notée "solution" dans le diagramme, est une solution non optimale. La seconde solution trouvée, notée "opt_sol" est la solution optimale pour l'instance. Les solutions et le site sont donc arrêtés car l'instance est résolue.

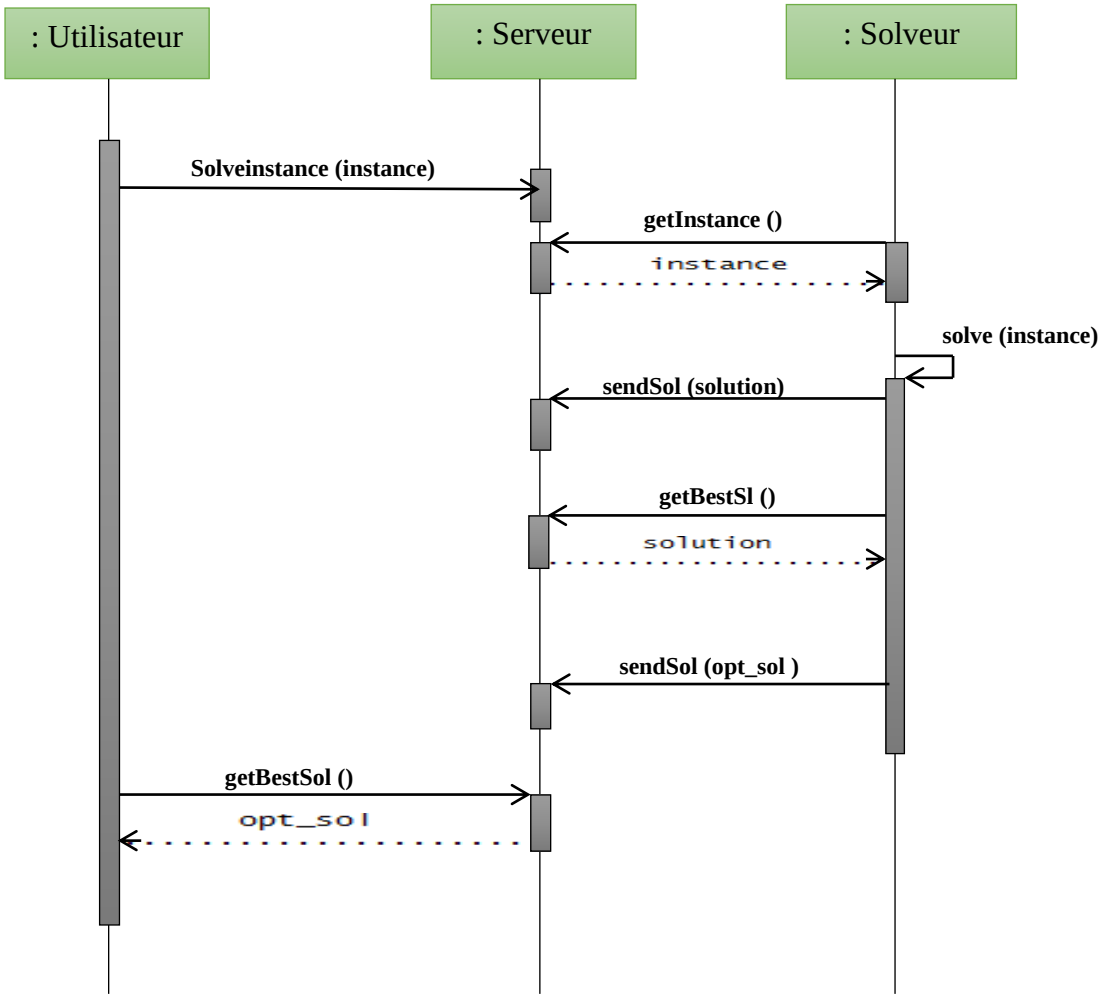


Figure 3.9 Diagramme de séquence d'une résolution d'instance du VRP avec la collaboration entre solutions.

Conclusion

Dans ce chapitre, nous avons passé à la conception détaillée pour expliquer les classes de notre application Ensuite, nous avons conçu quelques aspects dynamiques de notre application en se basant sur les diagrammes de séquence.

Dans le prochain chapitre, nous allons détailler l'implémentation et la réalisation du la solution.

Chapitre 04

IMPLEMENTATION DE PROJET

Introduction

Ce chapitre est consacré à la réalisation et l'implémentation de différentes fonctionnalités de notre programme. Nous expliquerons notre approche pour résoudre le problème de tournée de véhicules avec une capacité limitée. Nous avons choisi d'utiliser les algorithmes génétiques pour résoudre ce problème. Des résultats expérimentaux sont présentés afin de mesurer l'efficacité de notre solution.

4.1. L'objectif du travail

L'objectif de ce travail est l'utilisation d'une heuristique pour résoudre le problème de tournée de véhicules avec capacités limitées. La fonction Objectif consiste en la minimisation des coûts de transport, en termes de temps et de distance. Il consiste à visiter un nombre de villes (sites) en un minimum de distance sans visiter plusieurs fois par la même ville. Pour l'heuristique, nous avons opté pour les algorithmes génétiques.

4.2. Description de l'approche proposée

Pour résoudre ce problème, nous avons opté pour les algorithmes génétiques comme heuristique. Ce choix est justifié par les avantages suivants :

- Les algorithmes génétiques sont parmi les premières méthodes utilisées.
- Les algorithmes génétiques sont simples à implémenter.
- Les algorithmes génétiques permettent de traiter des espaces de recherche importants (beaucoup de solution, pas de parcours exhaustif envisagé).
- Pour un nombre de solution important, les algorithmes génétiques permettent d'éliminer les solutions non valides.
- Relativité de la qualité de la solution selon le degré de précision demandé..

4.3. Environnement matériel

L'implémentation de l'application va être réalisée sur une machine qui comporte les caractéristiques suivantes :

- Marque : Dell.
- Modèle : INSPIRON 15.
- Processeur : Intel® Core™ i5-3337U CPU © 1.80 GHz 1.80 GHz
- Mémoire installée (RAM): 4.00 Go.
- Type du système : Système d'exploitation 64bits.

- Système d'exploitation : Windows 10 édition professionnel

4.4 Environnement de développement

Ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels.

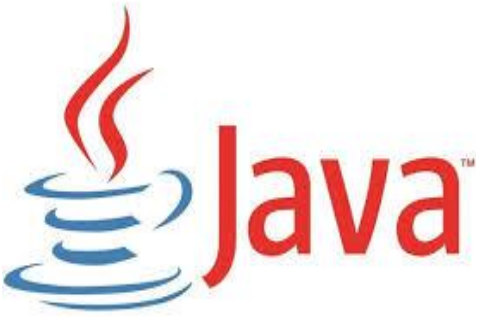
4.4.1. Langage de programmation

Pour l'implémentation de l'application, nous avons besoin de plusieurs outils et moyens à utiliser pour la création de l'application et faciliter le travail. Concernant le langage de programmation, il en existe plusieurs :

4.4.1.1. Java

La technologie Java définit à la fois un langage de programmation orienté objet et une plateforme informatique. Créée par l'entreprise Sun Microsystems (souvent juste appelée "Sun") en 1995, et reprise depuis par la société Oracle en 2009, la technologie Java est indissociable du domaine de l'informatique et du Web. On la retrouve donc sur les ordinateurs, mais aussi sur les téléphones mobiles, les consoles de jeux, etc. L'avènement du Smartphone et la puissance croissante des ordinateurs, ont entraîné un regain d'intérêt pour ce langage de programmation.

Depuis son lancement en 1995, Java a connu de multiples versions et améliorations. Ces dernières années, les versions Java 7 et Java 8, toutes deux existant en version Java 64 bits et 32 bits, font figure d'incontournables. Java 9 est actuellement en préparation. La technologie Java regroupe aujourd'hui différents standards, des logiciels et des communautés d'entreprises. Le terme est l'un des plus répandus sur la Toile et dans le monde de l'informatique. Il a même été utilisé comme symbole boursier au Nasdaq par Sun [4].



4.4.1.2. JSP

Signifie «Java Server Page». Cette norme a été développée par Sun Microsystems comme alternative à la technologie ASP (Active Server Page) de Microsoft. Les pages JSP sont similaires aux pages ASP en ce sens qu'elles sont compilées sur le serveur plutôt que dans le navigateur Web d'un utilisateur. Après tout, ils ne les appellent pas "pages de serveur" pour rien. Cependant, JSP est basé sur Java, tandis qu'ASP est basé sur Visual Basic. Les pages JSP sont utiles pour créer des sites Web dynamiques et accéder aux informations de base de données sur un serveur Web. Bien que les pages JSP puissent avoir Java entrecoupé de HTML, tout le code Java est analysé sur le serveur. Par conséquent, une fois que la page arrive dans le navigateur, ce n'est que du HTML. JavaScript, en revanche, est généralement analysé par le navigateur Web et non par le serveur Web [43].

4.4.1.3. Mysql

Prononcé soit «My SQL» ou «My Sequel», est un système de gestion de base de données relationnelle open source. Il est basé sur le langage de requête de structure (SQL), qui est utilisé pour ajouter, supprimer et modifier des informations dans la base de données. Les commandes SQL standard, telles qu'ADD, DROP, INSERT et UPDATE peuvent être utilisées avec MySQL.

MySQL peut être utilisé pour une variété d'applications, mais se trouve le plus souvent sur les serveurs Web. Un site Web qui utilise MySQL peut inclure des pages Web qui accèdent aux informations d'une base de données. Ces pages sont souvent appelées «dynamiques», ce qui signifie que le contenu de chaque page est généré à partir d'une base de données lors du chargement de la page. Les sites Web qui utilisent des pages Web dynamiques sont souvent appelés sites Web basés sur des bases de données.

De nombreux sites Web basés sur des bases de données qui utilisent MySQL utilisent également un langage de script Web comme PHP pour accéder aux informations de la base de données. Les commandes MySQL peuvent être incorporées dans le code PHP, permettant de générer une partie ou la totalité d'une page Web à partir des informations de la base de données. Parce que MySQL et PHP sont tous deux open source (ce qui

signifie qu'ils sont gratuits à télécharger et à utiliser), la combinaison PHP / MySQL est devenue un choix populaire pour les sites Web basés sur des bases de données [44].

4.4.1.4. Firebase

La base de données Firebase Realtime est une base de données NoSQL hébergée dans le cloud qui vous permet de stocker et de synchroniser vos utilisateurs en temps réel.
La base de données en temps réel n'est en réalité qu'un gros objet JSON que les développeurs peuvent gérer en temps réel.
Avec une seule API, la base de données Firebase fournit à votre application à la fois la valeur actuelle des données et toutes les mises à jour de ces données.

La synchronisation en temps réel permet à vos utilisateurs d'accéder facilement à leurs données depuis n'importe quel appareil, que ce soit Web ou mobile. Realtime Database aide également vos utilisateurs à collaborer les uns avec les autres.

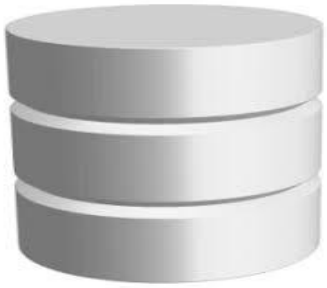
Un autre avantage incroyable de Realtime Database est qu'il est livré avec des SDK mobiles et Web, vous permettant de créer vos applications sans avoir besoin de serveurs. Lorsque vos utilisateurs se déconnectent, les SDK de base de données en temps réel utilisent le cache local sur l'appareil pour diffuser et stocker les modifications. Lorsque l'appareil est en ligne, les données locales sont automatiquement synchronisées. La base de données en temps réel peut également s'intégrer à l'authentification Firebase pour fournir un processus d'authentification simple et intuitif [45].

4.4.1.5 HTML

(HyperText Markup Language) est le bloc de construction le plus basique du Web. Il définit la signification et la structure du contenu Web. D'autres technologies que HTML sont généralement utilisées pour décrire l'apparence / la présentation (CSS) ou la fonctionnalité / le comportement d'une page Web (JavaScript).
«Hypertexte» fait référence aux liens qui relient des pages Web les unes aux autres, soit au sein d'un même site Web, soit entre des sites Web. Les liens sont un aspect

fondamental du Web. En téléchargeant du contenu sur Internet et en le reliant à des pages créées par d'autres personnes, vous devenez un participant actif du World Wide Web [46].

4.4.1.6. Système de Gestion de Base de Données (SGBD)



Le SGBD est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données.

Parmi les logiciels les plus connus il est possible de citer : MySQL, PostgreSQL, SQLite, Oracle Database, Microsoft SQL Server, Firebird ou Ingres [47].

4.4.1.7. CSS

Le terme CSS est l'acronyme anglais de *Cascading Style Sheets* qui peut se traduire par "feuilles de style en cascade". Le CSS est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou XML. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.

L'avantage de l'utilisation d'un fichier CSS pour la mise en forme d'un site réside dans la possibilité de modifier tous les titres du site en une seule fois en modifiants une seule partie du fichier CSS. Sans ce fichier CSS, il serait nécessaire de modifier chaque titre de chaque page du site (difficilement envisageable pour les énormes sites de plusieurs milliers de pages) [48].

4.4.1.8. Bootstrap

Est un framework CSS gratuit et open-source destiné au développement Web frontal réactif et mobile .

Il contient des modèles de conception CSS et (éventuellement) JavaScript pour la typographie , les formulaires , les boutons , la navigation et d'autres composants d'interface.

Bootstrap est le cinquième projet le plus étoilé sur GitHub , avec plus de 142 000 étoiles, derrière freeCodeCamp (près de 312 000 étoiles) et légèrement derrière le framework Vue.js. Selon Alexa Rank , Bootstrap est dans le top-2000 aux USA tandis que vuejs.org est dans le top-7000 aux USA [49].

4.4.1.9. C++

Est un langage de programmation à usage général créé par Bjarne Stroustrup comme une extension du langage de programmation C, ou "C avec des classes ". Le langage s'est considérablement développé au fil du temps, et le C ++ moderne dispose désormais de fonctionnalités orientées objet, génériques et fonctionnelles en plus des fonctionnalités de manipulation de la mémoire de bas niveau. Il est presque toujours implémenté comme un langage compilé, et de nombreux fournisseurs fournissent des compilateurs C ++, y compris la Free Software Fondation, LLVM, Microsoft, Intel, Oracle et IBM, il est donc disponible sur de nombreuses plates-formes[53].

4.4.1.10. CSP

C ++ Server Pages (CSP) est un moteur Web pour le développement d'applications Web avancées, qui utilise des scripts mixtes de langage de balisage / C ++ (tels que HTML / C ++, XML / C ++, WML / C ++, etc.)[54].

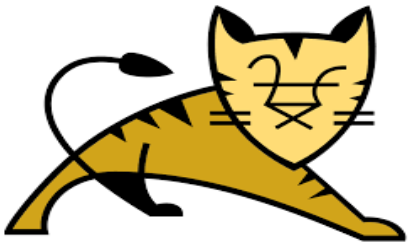
4.4.2. Les outils de développement

Un outil de développement est un logiciel qui aide un développé dans le déroulement d'une activité de développement.

4.4.2.1 Eclipse

Eclipse est un environnement de développement intégré (IDE) utilisé dans la programmation informatique . Il contient un espace de travail de base et un système de plug-in extensible pour personnaliser

l'environnement. Eclipse est écrit principalement en Java et son utilisation principale est pour développer des applications Java, mais il peut également être utilisé pour développer des applications dans d'autres langages de programmation via des plug-ins, notamment Ada , ABAP , C , C++ , C# , Clojure , COBOL , D , Erlang, Fortran , Groovy , Haskell , JavaScript , Julia , Lasso , Lua , NATURAL , Perl , PHP , Prolog , Python , R , Ruby (y compris le framework Ruby on Rails), Rust , Scala et Scheme . Il peut également être utilisé pour développer des documents avec LaTeX (via un plug-in TeXlipse) et des packages pour le logiciel Mathematica. Les environnements de développement incluent les outils de développement Java Eclipse (JDT) pour Java et Scala, Eclipse CDT pour C / C ++ et Eclipse PDT pour PHP, entre autres [50].



4.4.2.2. XAMPP

C'est un libre et open-source multi-plateforme serveur web pile de solution paquet développé par Apache Friends, composé essentiellement de le serveur HTTP Apache , la base de données MariaDB et les interpréteurs de scripts écrits dans les langages de programmation PHP et Perl . Étant donné que la plupart des déploiements de serveurs Web utilisent les mêmes composants que XAMPP, cela rend possible la transition d'un serveur de test local à un serveur en direct [51].

4.4.2.3. Apache Tomcat

C'est une implémentation open source des technologies Java Servlet, JavaServer Pages, Java Expression Language et Java WebSocket. Les spécifications Java Servlet, JavaServer Pages, Java Expression Language et Java WebSocket sont développées dans le cadre du Java Community Process .

Le logiciel Apache Tomcat est développé dans un environnement ouvert et participatif et publié sous la licence Apache version 2. Le projet Apache Tomcat est collaboration des meilleurs développeurs du monde entier [52].

4.5. Implémentation de la communication entre solutions

Liée à la mise en œuvre de projets

4.5.1. Architecture

Dans cette section, on va présenter l'architecture qui est utilisée pour faire communiquer les solutions entre eux durant les expériences. Sur le schéma suivant, on peut voir l'architecture globale :

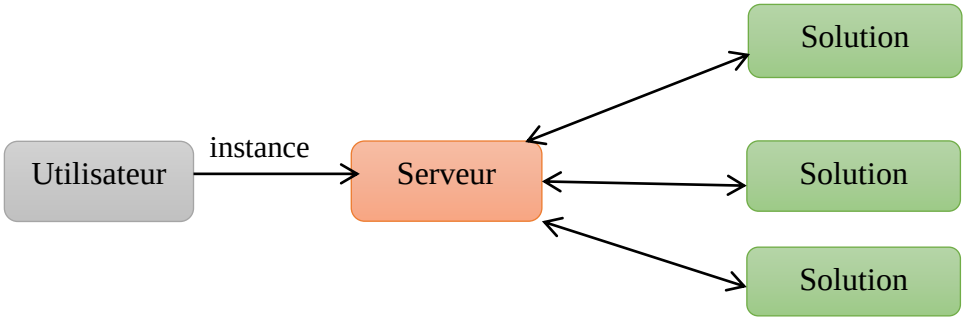


Figure 4.1 Architecture du protocole de communication.

- L'utilisateur est le programme qui servira à contrôler sur quelle instance les solutions travaillent.
- Il envoie à la solution le nom de l'instance que les solutions doivent résoudre et reçoit la solution optimale lorsque ceux-ci ont terminé. Si le programme est interrompu manuellement, la meilleure solution connue est renvoyée.
- Les solutions correspondent à ceux présentés dans le chapitre trois et qui ont été développés spécifiquement dans le cadre de ce travail pour permettre la collaboration. Sur le schéma, seul trois solutions sont indiqués, mais en pratique, un nombre arbitraire de solutions de chaque type peut être utilisé. Lorsque l'un de ceux-ci trouve 4une meilleure solution que la meilleure solution actuellement connue, il en informe le serveur qui pourra alors la transmettre aux autres solutions. Les solutions peuvent également être ajoutées dynamiquement, c'est-à-dire que l'on peut en connecter même lorsque la résolution d'une instance a déjà commencé. Les messages échangés ici sont uniquement des solutions complètes d'une instance de VRP, le but étant de rester générique afin de pouvoir utiliser ce protocole pour n'importe quel type de solution du VRP.

- La solution sert à établir la coordination entre les différents solveurs. C'est lui qui est chargé de propager les solutions connues et d'écouter le site pour déterminer quelle instance doit être résolue.

4.6. Expériences et résultats

Dans cette section, nous allons faire des expériences en utilisant les solutions pour résoudre des instances de différentes tailles afin de mesurer leurs performances. L'intérêt de ces expériences n'est pas les performances individuelles des solutions, mais bien la comparaison entre les performances avec ou sans collaboration. Chaque expérience a été répétée 5 fois. Les temps mesurés sont des temps réels et la machine n'était pas utilisée durant les expériences. Le processeur utilisé est un Intel® Core™ i5-3337U CPU © 1.80 GHz 1.80 GHz.

Dans les tableaux de résultats, la colonne "instance" fait référence au nom du fichier contenant l'instance. Les colonnes *n*, *K* et *C* sont respectivement le nombre de nœuds, le nombre de véhicules à utiliser et la capacité des véhicules.

4.6.1. Solution par optimisation par colonie de fourmis

Pour mesurer les performances de cette solution, la métrique utilisée sera le coût de la meilleure solution obtenue à la fin de l'exécution. Les paramètres de la solution utilisée sont :

- $n = 10000$, le nombre d'itérations
- *nrestarts* = 50, le nombre de redémarrage
- $\rho = 0.95$, le facteur d'évaporation
- $p = 0.25$, la proportion de sites considérés à chaque sélection d'une nouvelle destination
- $\alpha = \beta = 5$, indiquant la pondération entre les pistes de phéromones et la visibilité (voir équation 3.2)
- $Q = 1000000$, paramètre fixant l'échelle et n'ayant pas d'impact sur les performances.

Pour chaque instance testée, nous reporterons le meilleur, la moyenne et le pire coût obtenu après les 5 exécutions. Le coût de la solution optimale est également indiqué à titre indicatif. Le temps d'exécution moyen est aussi indiqué.

Cette solution est implémentée en JAVA.

Instance	*n*	*K*	*C*	meilleur	moyenne	pire
A-n37-k6	37	6	100	950, 85	959, 20	967, 51
A-n44-k7	44	7	100	953, 11	955, 88	963, 12
A-n45-k6	45	6	100	950, 22	956, 27	960, 69
A-n46-k7	46	7	100	925, 23	925, 65	925, 80
B-n43-k6	43	6	100	749, 03	750, 60	751, 79
B-n45-k5	45	5	100	770, 75	770, 75	770, 75
B-n56-k7	56	7	100	734, 99	737, 20	740, 35
E-n51-k5	51	5	160	524, 81	529, 28	536, 02
P-n50-k7	50	7	150	562, 64	564, 64	565, 68
P-n55-k7	55	7	150	579, 96	584, 43	589, 44

Table 4.1 Résultats des tests de performance de la méthode colonie de fourmis sur 10 instances différentes du VRP. Paramètres utilisés : n = 10000, *nrestarts* = 50, ρ = 0.95, p = 0.25, $\alpha = \beta = 5$, Q = 1000000.

La table **4.1** montre que cette solution est en mesure de fournir des solutions de qualité acceptable en un temps raisonnable, ce qui est le rôle de cette solution dans le cadre de la collaboration entre solutions. Pour l'instance **B-n56-k7, p = 0, 5** a été utilisé car la contrainte de capacité des véhicules était plus difficile à satisfaire, il fallait donc considérer un plus grand nombre de sites à chaque sélection de nouvelle destination pour la respecter.

5.6.2. Solution par optimisation linéaire en nombres entiers

Pour mesurer les performances de cette solution, nous utiliserons ici le temps d'exécution plutôt que le coût de la meilleure solution, puisque celui-ci trouve toujours la solution optimale (si l'exécution se termine sans interruption). Le temps mesuré est donc celui pris pour trouver la solution optimale. Nous reporterons le meilleur, la moyenne et le pire temps des 5 exécutions. Celui-ci est exprimé en secondes.

Cette solution est implémentée en utilisant JAVA.

Instance	*n*	*K*	*C*	meilleur	moyenne	pire
A-n37-k6	37	6	100	498, 77	503, 42	509, 46
A-n44-k7	44	7	100	518, 72	524, 29	528, 81
A-n45-k6	45	6	100	318, 24	320, 82	322, 15
A-n46-k7	46	7	100	171, 10	171, 67	172, 58
B-n43-k6	43	6	100	114, 64	115, 31	115, 76
B-n45-k5	45	5	100	60, 82	61, 07	61, 31
B-n56-k7	56	7	100	77, 57	78, 01	78, 46
E-n51-k5	51	5	160	68, 34	68, 87	69, 13
P-n50-k7	50	7	150	112, 27	113,00	113, 73
P-n55-k7	55	7	150	5298,70	5346,39	5412, 88

Table 4.2 Comparaison entre les tests de performance de la solution ILP pour la résolution de 10 instances différentes du VRP.

Comme nous pouvons le voir dans la table 4.2, la collaboration a globalement un impact mitigé sur les performances. Les performances obtenues dans le meilleur des cas sont toujours améliorées. Celles obtenues dans le cas moyen sont parfois améliorées, et de même pour le pire cas.

Il semblerait que la collaboration avec cette configuration soit la plus effective pour des instances étant résolues en un temps modéré, entre 10 secondes et 1 minute. Une interprétation possible est que dans le cas d'instances faciles à résoudre, la collaboration n'a pas le temps d'être efficace et n'apporte donc rien. Dans le cas d'instances difficiles, il est très probable qu'au bout d'un long temps d'exécution les solutions ACO ne parviennent plus à améliorer leur meilleure solution alors qu'ils consomment encore beaucoup de ressources.

Testons maintenant une deuxième configuration qui devrait être plus adaptée pour les instances plus difficiles à résoudre (celles prenant au minimum 60 secondes pour être résolue sans collaboration). Cette configuration consiste à utiliser quatre solutions ACO pendant 10 secondes afin de trouver rapidement une bonne solution, puis les déconnecter et laisser place à la solution ILP. On utilise donc cette configuration pendant 10 secondes :

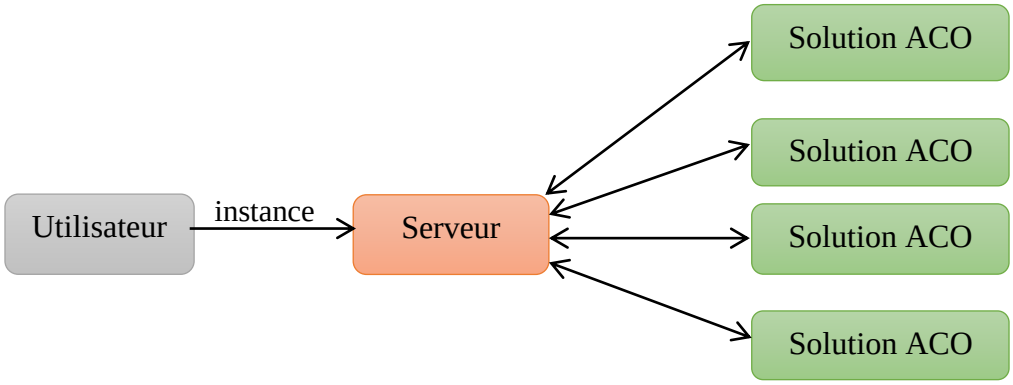


Figure 4.2 Première partie de la deuxième configuration utilisée pour les expériences de collaboration entre solutions : quatre solutions ACO pendant 10 secondes.

Puis cette configuration jusqu'à la fin de la résolution :

Figure 4.3 Deuxième partie de la deuxième configuration utilisée pour les expériences de collaboration entre solutions : une solution ILP jusqu'à résolution de l'instance.

4.6.3 Collaboration de deux méthodes

Le but de cette configuration est donc de laisser du temps aux solutions ACO de trouver une bonne solution pour que la solution ILP puisse ensuite s'en servir comme borne. Les temps d'exécution reportés sont la somme des 10 secondes utilisées par les solutions ACO et le temps d'exécution de la résolution avec la solution ILP

Instance	*n*	*K*	*C*	meilleur	moyenne	pire
A-n37-k6	37	6	100	219, 83	317, 92	505, 12
A-n44-k7	44	7	100	215, 03	308, 53	364, 74
A-n45-k6	45	6	100	277, 96	326, 46	371, 76
A-n46-k7	46	7	100	213, 09	334, 43	444, 78
B-n43-k6	43	6	100	65, 30	75, 87	89, 83
B-n45-k5	45	5	100	31, 51	43, 28	73, 78
B-n56-k7	56	7	100	41, 61	69, 63	109, 53
E-n51-k5	51	5	160	44, 20	55, 28	68, 02
P-n50-k7	50	7	150	85, 18	100, 14	113, 9
P-n55-k7	55	7	150	2029, 71	3825, 03	5521, 26

Table 4.3 Résultats des tests de performance lors de la collaboration entre solutions pour la résolution de 10 instances différentes du VRP. Configuration utilisée : quatre solutions ACO pendant 10 secondes puis une solution ILP.

On peut voir dans la table 4.3 que la variance dans les temps d'exécution est également présente, pour la même raison que dans notre première configuration. Faisons maintenant à nouveau la comparaison avec les résultats obtenus pour la solution ILP sans collaboration.

Instance	*n*	*K*	*C*	meilleur	moyenne	pire
A-n37-k6	37	6	100	44,1%	63,1%	99,2%
A-n44-k7	44	7	100	41,5%	58,9%	69,0%
A-n45-k6	45	6	100	87,3%	101,8%	115,4%
A-n46-k7	46	7	100	124,5%	194,8%	257,7%
B-n43-k6	43	6	100	57,0%	65,8%	77,6%
B-n45-k5	45	5	100	51,8%	70,9%	120,3%
B-n56-k7	56	7	100	53,6%	89,3%	139,6%
E-n51-k5	51	5	160	64,7%	80,3%	98,4%
P-n50-k7	50	7	150	75,9%	88,6%	100,2%
P-n55-k7	55	7	150	38,3%	71,5%	102,0%

Table 4.4 Collaboration entre solutions pour la résolution de 10 instances différentes du VRP. Configuration utilisée lors de la collaboration entre solutions : quatre solutions ACO pendant 10 secondes puis une solution ILP.

Les résultats de la table 4.4 montrent une amélioration des performances pour certaines instances et une dégradation pour d'autres, par rapport à ceux de la table 5.3. On notera aussi que cette configuration est bien moins gourmande en ressources, puisque la majorité du temps d'exécution se déroule avec une seule solution active.

Cette deuxième expérience montre que le choix de la configuration utilisée pour la collaboration entre solutions est un élément capital pour obtenir de bonnes performances, certaines configurations. Sont mieux adaptées que d'autres pour certaines instances.

Conclusion

Ces expériences ont permis de montrer que la collaboration entre les solutions obtient de bons résultats mais nécessite peut-être un raffinement pour être plus performante. Je recommanderais de développer un algorithme dédié à la gestion de la configuration de solutions utilisées. Pour ces expériences, la configuration était gérée manuellement par l'utilisateur. Un algorithme dédié pourrait quant à lui gérer quand tel ou tel solution doit être utilisé ou éteint. En début d'exécution, il pourrait par exemple lancer un grand nombre de solutions trouvant des bonnes solutions rapidement, tel que des solutions par recherche locale ou des solutions ACO. Il pourrait ensuite les éteindre au fur et à mesure afin de libérer des ressources pour la solution ILP.

CONCLUSION GENERALE

Dans ce travail, nous avons vu en quoi consistait le problème de tournées de véhicules hétérogènes à capacités limitées avec ramassage, comment celui-ci a été étendu à de nombreuses variantes pour tenir compte des contraintes de la vie réelle et les différentes approches qui sont utilisées pour le résoudre. Un état de l'art de ces différentes méthodes a été fait afin de voir quelles techniques étaient les mieux adaptées pour résoudre ce problème. Parmi celles-ci, deux ont été retenues et ont été implémentés pour être utilisées lors de la collaboration entre solution.

Le premier solution implémenté, le solution par optimisation par colonie de fourmis, a permis de montrer comment le comportement d'une colonie de fourmis pouvait être exploité pour développer un algorithme d'optimisation. Le rôle de cette solution a été de fournir de bonnes solutions aux autres solutions lors de la collaboration, les tests de performance ont pu montrer qu'il remplissait bien son rôle.

Ensuite, nous avons vu un deuxième type de solution, la solution par optimisation linéaire en nombres entiers. Celui-ci se base sur le modèle mathématique du problème pour le résoudre. Ceci a permis de montrer comment un algorithme de résolution de problème ILP fonctionnait et comment il pouvait être utilisé pour résoudre le VRP. Le rôle de ce solution était de résoudre l'instance du problème optimale ment en utilisant les solutions de bonne qualité fournies par les autres solutions.

Enfin, nous avons fait collaborer ces deux différentes solutions entre eux. Les expériences ont pu montrer que l'impact de cette collaboration était globalement positif, mais que cela introduisait aussi une grande variance dans les résultats, en ayant parfois des temps d'exécution presque doublés d'une exécution à une autre. Ces résultats semblent indiquer qu'il y a un réel intérêt à faire collaborer différents types de solutions entre eux.

Bibliographies

[1] Barthélemy, Jean-Pierre, and François Brucker. "NP-hard approximation problems in overlapping clustering." Journal of classification 18.2 (2001): 159-183.

[2] Charon, Irène, Anne Germa, and Olivier Hudry. Méthodes d'optimisation combinatoire. Paris: Masson, 1996.

[3] Monge, Gaspard. Géométrie descriptive. J. Klostermann fils, 1811.

[4] Fichtenholz, Grigorii, and Leonid Kantorovitch. "Sur les opérations linéaires dans l'espace des fonctions bornées." Studia Mathematica 5.1 (1934): 69-98.

[5] Dorigo, Marco, and Luca Maria Gambardella. "Ant colonies for the travelling salesman problem." biosystems 43.2 (1997): 73-81.

[6] Charbonnier, Christine. Le routage dynamique des vehicules. Diss. École nationale supérieure de l'aéronautique et de l'espace (Toulouse; 1972-2007), 1991.

[7] Charbonnier, Christine. Le routage dynamique des vehicules. Diss. École nationale supérieure de l'aéronautique et de l'espace (Toulouse; 1972-2007), 1991.

[8] BICKEL, Peter J., et al. Simultaneous analysis of Lasso and Dantzig selector. The Annals of statistics, 2009, 37.4: 1705-1732.

[9] Akli, Meriem. Problème de tournées de véhicules avec contraintes et fenêtre de temps. Diss. UMMTO, 2013.

[10] wikipedia,https://fr.wikipedia.org , consulté le : 05/05/2019.

[11] Laporte, G. (1992). The vehicle routing problem: European journal of operational research, 59(3), 345-358.

[12] Rego, C., & Roucairol, C. (1994). Le problème de tournées de vehicules: Étude et résolution approchée.

[13] Laporte, Gilbert, Paolo Toth, and Daniele Vigo. "Vehicle routing: historical perspective and recent contributions." (2013): 1-4

[14] Brotcorne, Luce, Gilbert Laporte, and Frederic Semet. "Ambulance location and relocation models." European journal of operational research 147.3 (2003): 451-463.

[15] Drouche, Hakima. Application du simplexe classique de Dantzig à un problème linéaire flou. Diss. UMMTO, 2016.
[16] Gärtner, Bernd, Martin Henk, and Günter M. Ziegler. "Randomized simplex algorithms on Klee-Minty cubes." Combinatorica 18.3 (1998): 349-372
[17] Madani, Amel. Le problème de transport des personnes dans les sociétés de taxis. Diss. FACULTE MATHEMATIQUES ET INFORMATIQUE DEPARTEMENT INFORMATIQUE-DOMAINE: Mathématiques Et Informatique FILIERE: Informatique OPTION: IDO, 2019.
[18] Akoa, François Bertrand. Approches de points intérieurs et de programmation DC en optimisation non convexe. Code et simulations numériques industrielles. These de Doctorat de l'Université de Rouen, 2005.
[19] Hao, J. K., Galinier, P., & Habib, M. (1999). Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. Revue d'intelligence artificielle, 13(2), 283-324.
[20] Basseur, Matthieu. Conception d'algorithmes coopératifs pour l'optimisation multi-objectif: application aux problèmes d'ordonnancement de type flow-shop. Diss. Lille 1, 2005.
[21] Kaufmann, A. "La programmation dynamique et ses possibilités en calcul économique." JORBEL-Belgian Journal of Operations Research, Statistics, and Computer Science 3.3-4 (1963): 64-74.
[22] Madani, Amel. Le problème de transport des personnes dans les sociétés de taxis. Diss. FACULTE MATHEMATIQUES ET INFORMATIQUE DEPARTEMENT INFORMATIQUE-DOMAINE: Mathématiques Et Informatique FILIERE: Informatique OPTION: IDO, 2019.
[23] Akli, M. (2013). Problème de tournées de véhicules avec contraintes et fenêtre de temps (Doctoral dissertation, UMMTO).
[24] CHRISTOFIDES, Nicos; MINGOZZI, Aristide; TOTH, Paolo. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical programming, 1981, 20.1: 255-282.
[25] HAMILTON, Carol. AAAI News. AI Magazine, 2002, 23.1: 5-5.
[26] E. Grellier, Optimisation des tournées de véhicules dans le cadre de la logistique inverse : modélisation et résolution par des méthodes hybrides, Thèse de doctorat de l'université de Nantes ,2008.

[27] GRAHAM, Ronald L., et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics. Elsevier, 1979. p. 287-326.
[28] Dorigo M. et Stützle T. Ant Colony Optimization. MIT Press, 2004, p. 155–159.
[29] Algorithme de colonies de fourmis. url : https://fr.wikipedia.org/wiki/Algorithme_ de_colonies_de_fourmis (visité le 25/02/2018).
[30] Altınel, İ. K., and Temel Öncan. "A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem." Journal of the Operational Research Society 56.8 (2005): 954-961.
[31] Paessens H. « The savings algorithm for the vehicle routing problem ». In : European Journal of Operational Research 34.3 (1988), p. 336–344.
[32] Russel S. et Norvig P. Artificial Intelligence : A Modern Approach. 3e éd. Pearson Education, 2003.
[33] Legrand, Thomas, et al. "Stratégies mimétiques pour les essaims de particules et implémentation sous ParadisEO." Second séminaire francophone sur l'optimisation par essaim particulaire (OEP) (2007).
[34] Yiyong X. et al. « Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems ». In : Engineering Optimization 46.4 (2013), p. 562–579.
[35] Gendreau M., Hertz A. et Laporte G. « A Tabu Search Heuristic for the Vehicle Routing Problem ». In : Management Science 40.10 (1994), p. 562–579.
[36] Lin S. et Kernighan B.W. « An Effective Heuristic Algorithm for the Traveling Salesman Problem ». In : Operations Research 21.2 (1973), p. 498–516.
[37] Yu B., Yang Z.-Z. et Yao B. « An improved ant colony optimization for vehicle routing problem ». In : European Journal of Operational Research 196.1 (2009), p. 171–176.
[38] Integer programming. url : https://en.wikipedia.org/wiki/Integer_programming (visité le 07/05/2018).
[39] Convex hull. url : https://en.wikipedia.org/wiki/Convex_hull (visité le 10/05/2018).
[40] Laporte G., Nobert Y. et Desrochers M. « Optimal Routing under Capacity and Distance Restrictions ». In : Operations Research 33.5 (1985), p. 1050–1073.
[41] Lourenço, Sergio O., et al. "Amino acid composition, protein content and calculation of nitrogen-to-protein conversion factors for 19 tropical seaweeds." Phycological Research 50.3 (2002): 233-241.

[42] Java, Akshay, et al. "Why we twitter: understanding microblogging usage and communities." Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis. 2007.
[43] Liu, Chien-Hung. "Data flow analysis and testing of JSP-based Web applications." Information and Software Technology 48.12 (2006): 1137-1147.
[44] Egea, Marina, Carolina Dania, and Manuel Clavel. "MySQL4OCL: A stored procedure-based MySQL code generator for OCL." Electronic Communications of the EASST 36 (2010).
[45] Cheng, Fu, and Fu Cheng. Build Mobile Apps with Ionic 4 and Firebase. Apress, 2018.
[46] World Wide Web Consortium. "HTML 4.01 specification." (1999).
[47] Morisset, Charles. Sémantique des systèmes de contrôle d'accès: définition d'un cadre sémantique pour la spécification, l'implantation et la comparaison de modèles de contrôle d'accès. Diss. Paris 6, 2007.
[48] Kan, D. M. (1958). On homotopy theory and css groups. Annals of Mathematics, 38-53.
[49] Hall, Peter. "Methodology and Theory for the Bootstrap." Handbook of econometrics 4 (1994): 2341-2381.
[50] Biermann, E., Ehrig, K., Köhler, C., Kuhns, G., Taentzer, G., & Weiss, E. (2006, October). Graphical definition of in-place transformations in the eclipse modeling framework. In International Conference on Model Driven Engineering Languages and Systems (pp. 425-439). Springer, Berlin, Heidelberg.
[51] Rubel, Sk, and K. Venkatesh. "Software defined tool virtual cloud, which help to setup, manage, and audit virtual data center environment." 2017 International Conference on Inventive Computing and Informatics (ICICI). IEEE, 2017.
[52] "Apache Tomcat - Welcome!". Apache Tomcat. The Apache Software Foundation. 2020-09-15. Retrieved 2020-09-16.
[53] Meyers, Scott. Effective C++: 55 specific ways to improve your programs and designs. Pearson Education, 2005.
[54] Brodsky, Stephen Andrew, Shyh-mei F. Ho, and II James Rush Rhyne. "Common application metamodel including C/C++ metamodel." U.S. Patent No. 7,275,079. 25 Sep. 2007.

Annexes

1. Variantes du VRP

La variation des paramètres du VRP, la suppression et/ou l'ajout ou bien la combinaison de contraintes du VRP classique permettent de définir un ensemble de variantes du VRP. Dans ce qui suit, nous présentons une répartition des variantes du VRP selon le type de contraintes.

1.1 VRP-C (Vehicle Routing Problem with Capacitated)

C'est un problème de tournées de véhicules avec des contraintes de capacités.

Les véhicules ont une capacité d'emport limitée (quantité, volume, poids,. . .). Ceci se traduit par le fait que la somme des demandes des clients appartenant à une tournée ne doit pas dépasser la capacité du véhicule. En pratique tous les problèmes de tournées de véhicules sont à capacité limitée seulement le degré d'influence de cette contrainte varie. Par exemple, les problèmes de transport de marchandise sont très sensibles à la capacité alors que le problème de distribution de courrier l'est moins.

1.1.1 VRP-FL (Vehicle Routing Problem with Full Truckload)

C'est un VRP avec utilisation complète de la capacité du véhicule.

1.1.2 VRP-HF (Vehicle Routing Problem with Heterogeneous Fleet)

Pour ce problème la flotte est composée de véhicules de types différents, qui se distinguent par la capacité, la puissance, le cout de transport, . . .

1.1.3 O-VRP (Open Vehicle Routing Problem)

Ce problème est identique au VRP, seulement le véhicule est libre de rejoindre ou pas le dépôt après la fin de la tournée. S'il choisit de reprendre le dépôt, il doit reprendre le parcours de la tournée dans le sens inverse.

1.1.4 VRP-B (Vehicle Routing Problem Back)

Le VRP-B est un problème ou le retour du véhicule au dépôt est exigé. Le véhicule doit rejoindre le dépôt aussitôt que le dernier client est servi sans reprendre le parcours de la tournée.

1.2 VRP a Contraintes Liées à la Demande des Clients

La demande des clients est une donnée importante car c'est elle qui va régir le déclenchement des livraisons.

1.2.1 VRP a Demande Déterministe

C'est un problème fréquent pour les entreprises qui font des livraisons sur commande. En effet, le livreur connait avant son départ du dépôt la quantité à livrer à chacun de ses clients.

1.2.2 VRP a Demande Stochastique

Contrairement au précédent, dans le VRP a demandé stochastique, le livreur ne connait pas la quantité à livrer au client il la découvre au moment de le servir. Il estime approximativement la demande de chaque client par une fonction stochastique.

1.2.3 S-VRP (Split Delivery Vehicle Routing Problem)

Ce problème consiste à visiter un client plusieurs fois afin de satisfaire entièrement sa demande. Exceptionnellement pour ce problème, la demande du client peut être supérieure à la capacité du véhicule.

1.2.4 VRP-PD (Vehicle Routing Problem Pick-up and Deliveries)

C'est un problème de tournées de véhicules avec collecte et livraison. Avec ce genre de problème la durée du service est comptabilisée deux fois, car on doit effectuer une collecte et une livraison ou inversement.

1.3 VRP a Contraintes Liées aux Dépôts

La capacité du dépôt est aussi importante car c'est elle qui va permettre de déterminer la quantité totale livrable. Elle peut être limitée ou non.

1.3.1 VRP-MD (Multi-Dépôt Véhicule Routing Problem)

Les véhicules peuvent s'approvisionner de plusieurs dépôts.

1.3.2 VRP-SD (Single-Depot Vehicle Routing Problem)

Les véhicules doivent s'approvisionner d'un seul dépôt. On peut passer du VRP-MD au VRP-1D ou inversement par centralisation respectivement division du dépôt(s).

1.4 VRP à Contraintes Liées aux Produits

Le produit peut se présenter par son mode de fabrication soit à la pièce soit en mode continue

1.4.1 MP-VRP (Multiple Product Vehicle Routing Problem)

Une gamme de produits doit être livrée aux différents clients par chaque véhicule en plusieurs tournées.

1.4.2 SP-VRP (Single Product Vehicle Routing Problem)

Un seul produit doit être livre aux différents clients par chaque véhicule en une seule Tournée.

1.5 VRP à Contraintes Liées au Temps

Les fenêtres de visite ou fenêtres de temps correspondent à un intervalle de temps pendant lequel une visite chez le client est possible.

1.5.1 PVRP (Periodic Vehicle Routing Problem)

Dans le problème de tournées de véhicules périodique, chaque client est périodiquement visite selon une certaine planification prédéfinie.

1.5.2 VRP à Temps de Service Déterministe

La durée du service est connue par le livreur avant d'entamer la tournée.

1.5.3 VRP à Temps de Service Stochastique

Le livreur ne connait pas la durée du service des clients, il l'a découvre au moment de les servir. Il peut définir une fonction stochastique pour l'approximer.

1.5.4 VRP-TW (Vehicle Routing Problem with Time Windows)

Le VRPTW est un problème de tournées de véhicules avec fenêtre de temps. Chaque client doit être servi dans un intervalle de temps défini, connu d'avance par le livreur et toute violation de cette contrainte peut engendrer une pénalité. Lorsque la contrainte de fenêtre de temps n'est pas satisfaite, soit on rejette la solution si on considère le cas rigide ou bien on construit une fonction de pénalité qui sera rajoutée ou combinée avec la fonction objective

pour le cas relâche. En réalité, c'est un problème très fréquent. La distribution des produits périssables (le lait, la viande, . . .), de journaux, services ambulatoires, . . . sont des exemples pratiques du VRP-TW. Dans cette classe de problèmes, on distingue deux sous-classes :

- le VRP-TW rigide ou le service doit impérativement être effectue dans la fenêtre de temps.
- le VRP-TW relâche où le retard ou l'avance engendre uniquement une pénalité. Lorsqu'on rajoute on définit une variante du VRPTW. Nous citons le CVRPTW, le PDPTW, le 1-PDPTW, le m-PDPTW, etc.

1.6 VRP Statique (Static Vehicle Routing Problem)

Le VRP Statique est un problème dont toutes les composantes sont connues, fixées avant d'entamer la moindre tournée. Ce cas est obtenu soit par une étude rigoureuse de actionnarisation de l'ensemble des paramètres soit parce que le problème modélise à l' origine un phénomène statique.

1.7 VRP dynamique (Dynamic Vehicle Routing Problem)

A l'opposé du VRP Statique, le VRP Dynamique a au moins une composante dynamique ou changeante au cours de son exécution. On peut avoir la demande qui ou la fenêtre de temps de servitude qui varient ou bien le nombre de clients à servir qui change. Dans ce cas, le problème devient plus complexe. [26].

2 Quelques images du site

2.1 Authentification

L'utilisateur doit remplir les champs pour accéder au site.

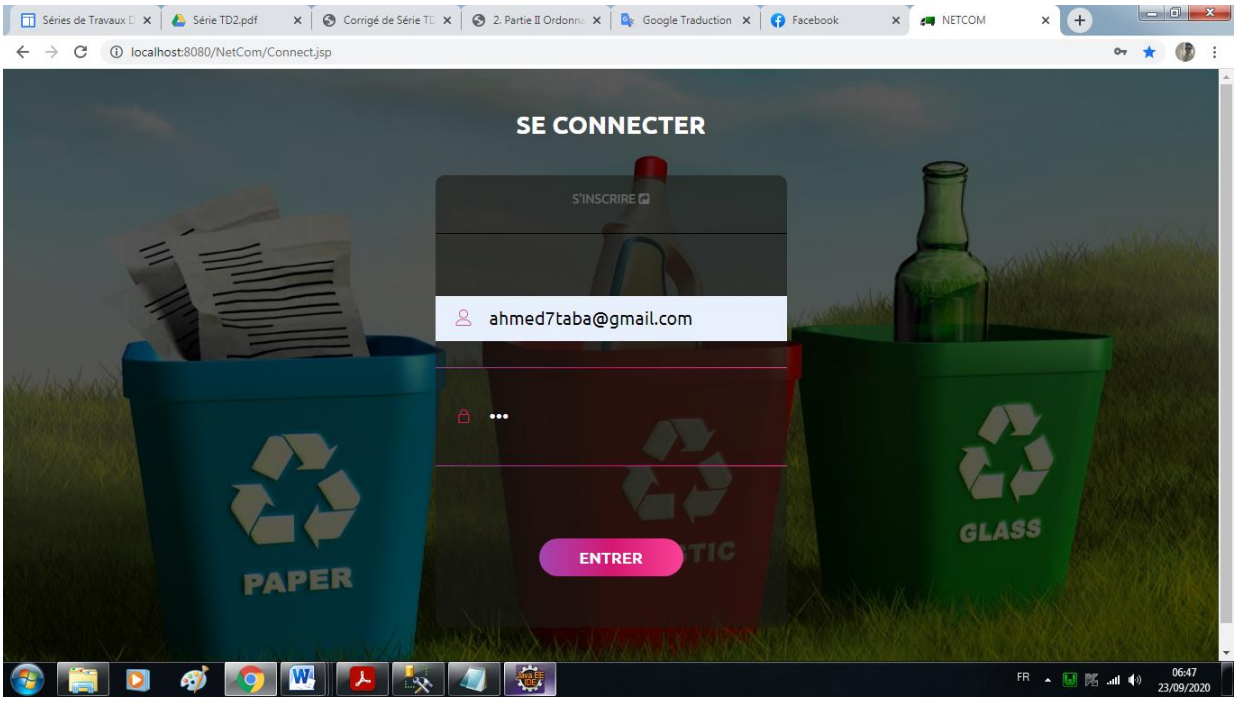


Figure 2.1 Authentification.

2.2 Véhicule

L'administrateur est capable de :

- Ajouter un véhicule.
- Recherche par marque.
- Modifier les caractéristiques du véhicule.
- Supprimer un véhicule.
- Ajouter une contrainte.

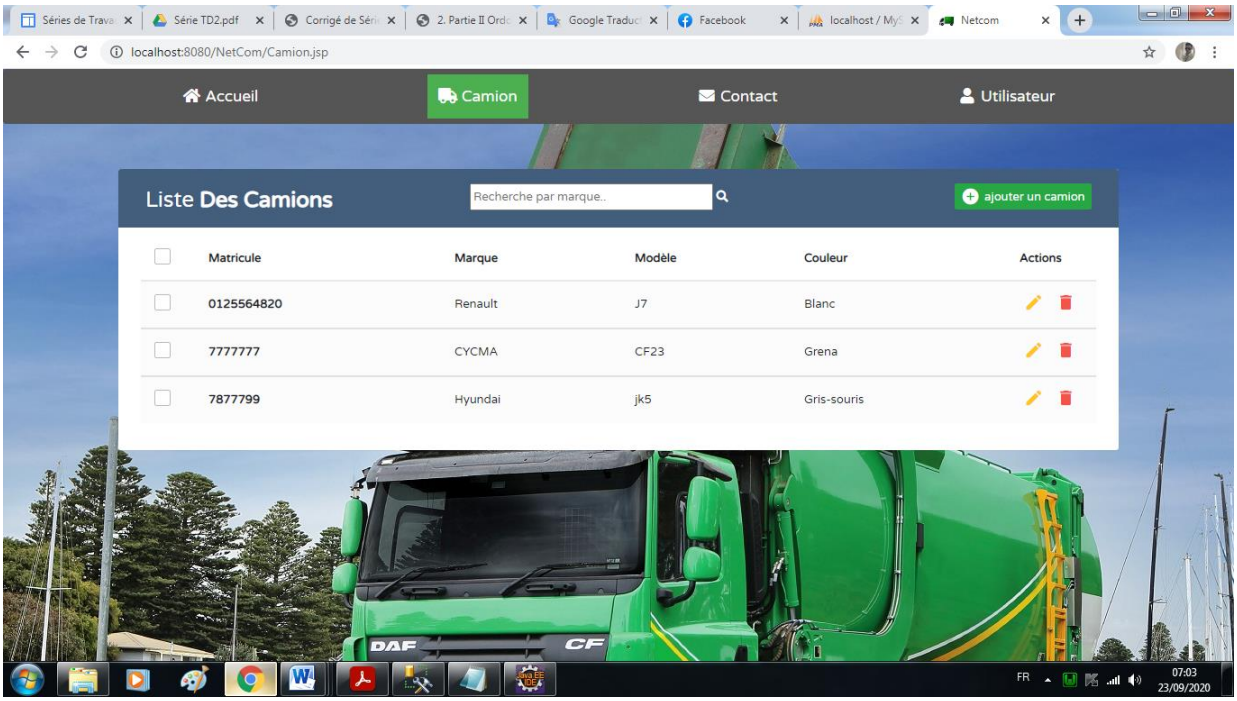


Figure 2.2 Liste des Véhicules

5.9.3. Afficher le chemin sur map

Après avoir introduit les différentes informations relatives à chaque tâche de ramassage au niveau des m sites, il faut afficher le plus court chemin sur la carte (googlemap ou autres) et sélectionner les véhicules selon leurs capacités.

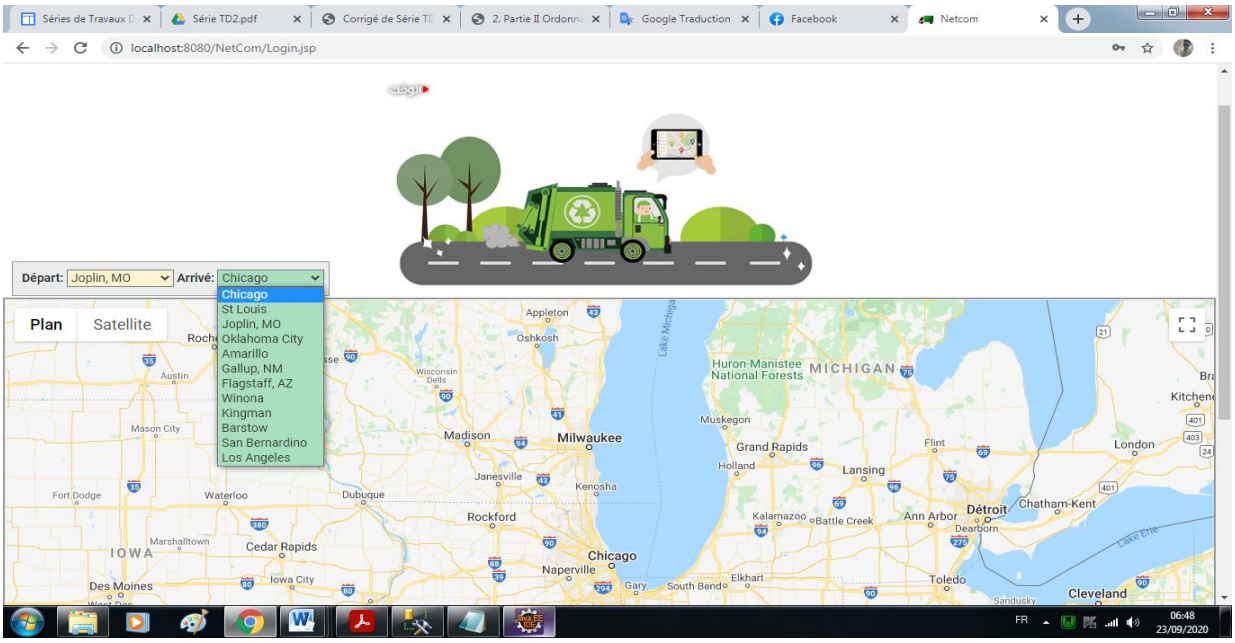


Figure 2.3 Map avant de indiquant le chemin à suivre pour atteindre le site.