N° d'ordre :09/2008 – M /MT

الجمهورية الجزائرية الديمقراطية الشعبية
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

وزارة التعليم العالي والبحث العلمي

جامعة هواري بومدين للعلوم والتكنولوجيا

Faculté de Mathématiques

Mémoire

Présenté pour l'obtention du diplôme de MAGISTER

En : **Mathématiques**

Spécialité : **Recherche Opérationnelle : Mathématiques Discrètes et Optimisation**

Par : **KHEFFACHE Rezika**

Thème

Scatter Search pour le problème de tournées de véhicules

Soutenu le 23 Septembre 2008, devant le jury composé de :

Président	**Abdelhafid BERRACHEDI**	**Professeur**	**U.S.T.H.B.**
Rapporteur	**Rachid OUAFI**	**Maître de Conférences**	**U.S.T.H.B.**
Examinateur	**Hacène AIT HADDADENE**	**Professeur**	**U.S.T.H.B.**
Invité	**Mohamed YAGOUNI**	**Chargé de Cours**	**U.S.T.H.B.**

A la mémoire de mon père

A ma mère

A mes soeurs : Nadia, Hamida, Lylia, Karima, Ouzna et Ouardia.

A mes neveux : Samy, Belkacem et Saîd.

A mes beaux frères : Slimane, Mohand et Nabil.

Remerciements

Mes remerciements les plus vifs et chaleureux, empreints d'une reconnaissance ineffable vont à mon directeur de mémoire monsieur **OUAFI Rachid**,*Maître de conférences à* **USTHB**, *pour ses précieux conseils, ses remarques pertinentes et son entière disponibilité.*

Je désire exprimer ma gratitude à monsieur **BERRACHEDI Abdelhafid**, *Professeur à* **USTHB**, *pour l'honneur qu'il me fait d'avoir accepté de présider le jury de soutenance.*

Mes sincères remerciements s'adressent également à monsieur **AIT HADDADENE Hacène**, *Professeur à* **USTHB** *pour avoir accepté de participer au jury.*

Je remercie tout particulièrement monsieur **YAGOUNI Mohamed**, *Chargé de cours à* **USTHB**, *pour son aide et l'analyse minutieuse qu'il a menée sur le manuscrit, qu'il en soit humblement remercié.*

Ce mémoire n'aurait pas pu s'écrire sans l'appui moral de ma famille, de mes amis et de mes collègues. Je les remercie tous et souhaite que la lecture qui s'offre à leur curiosité leur procure la satisfaction qu'ils espéraient.

Table des matières

Résumé **3**

Introduction générale **4**

1 Optimisation Combinatoire et Métaheuristiques **7**
1.1 Introduction . . . 7
1.2 Optimisation Combinatoire . . . 7
1.3 Complexité . . . 7
1.3.1 Algorithme efficace . . . 8
1.3.2 Les classes des problèmes . . . 8
1.4 Méthodes exactes . . . 9
1.5 Méthodes approchées . . . 9
1.5.1 Heuristiques . . . 10
1.5.2 Métaheuristiques . . . 10

2 Le problème de tournées de véhicules **15**
2.1 Introduction . . . 15
2.2 Le problème de tournées de véhicules et ses généralisations . . . 16
2.3 Modélisation . . . 18
2.3.1 Le problème de voyageur de commerce . . . 18
2.3.2 Le problème des m voyageurs de commerce . . . 19
2.3.3 Le problème de tournées de véhicules . . . 20
2.4 Complexité du VRP . . . 23
2.5 Les méthodes de résolution . . . 23
2.5.1 Méthodes exactes . . . 23
2.5.2 Méthodes approchées . . . 25

3 Scatter Search **31**
3.1 Définition et origine . . . 31
3.1.1 Principes de Scatter search . . . 32
3.2 Procédure de Scatter search . . . 32
3.2.1 Méthode de génération de diversification . . . 33
3.2.2 Ensemble de solutions de référence . . . 34
3.2.3 Méthode de génération des sous ensembles . . . 35
3.2.4 Méthode de combinaison des solutions . . . 35
3.3 Algorithme de Scatter search . . . 36

3.4 Evolution de Scatter Search et applications 38
3.4.1 Evolution de Scatter Search 38
3.4.2 Applications 39

4 Scatter Search pour le problème de tournées de véhicules 43
4.1 Introduction 43
4.2 Méthode de génération de diversification 45
4.3 Méthode d'amélioration 47
4.4 Méthode de mise à jour de l'ensemble de référence 49
4.5 Méthode de génération de sous ensembles 49
4.6 Méthode de combinaison 50
4.7 Méthode d'amélioration 53

5 Scatter Search pour le problème de tournées de véhicules 57
5.1 Introduction 57
5.2 Procédure de Scatter search 58
5.2.1 Méthode de génération de diversification 58
5.2.2 Méthode d'amélioration 59
5.2.3 Méthode de mise à jour de l'ensemble de référence 59
5.2.4 Méthode de génération des sous ensembles 59
5.2.5 Méthode de combinaison 60
5.3 Algorithme de Scatter Search pour le problème de tournées de véhicules . . 61
5.4 Exemple d'illustration 64

Conclusion et perspectives 74

Bibliographie 75

Résumé

L'approche évolutive appelée "Scatter search" ou "recherche dispersée" provient de recherche des stratégies pour la création de règle de décision et de contraintes de substitution.

Des études récentes montrent les avantages pratiques de cette approche pour résoudre divers problèmes d'optimisation.

Notre objectif est d'illustrer comment cette méthode peut être utilisée efficacement pour la résolution des problèmes de permutation qui impliquent la détermination optimale de cycles (ou circuits) en théorie des graphes et optimisation combinatoire.

Dans notre mémoire, nous identifions une conception générale pour résoudre le problème de tournées de véhicules.

Mots clés : Métaheuristiques, scatter search, optimisation combinatoire, tournées de véhicules.

Introduction générale

Le développement technologique soumet l'Homme aux contraintes d'un système de relations économiques de plus en plus complexes. Les chercheurs constatent qu'il y a de plus en plus d'éléments nouveaux qui doivent être pris en compte lors des prises de décision concernant une action donnée (organisation d'une production, un réseau de transport...etc.) et que ces prises de décision deviennent l'objet de véritables recherches qui ne peuvent être menées sans l'aide d'outils mathématiques. C'est ainsi que s'est développé un domaine des mathématiques basé sur l'activité de décision, appelé Recherche Opérationnelle [2].

Les premiers problèmes qui marquent le début de la Recherche Opérationnelle ont été posés pendant la seconde guerre mondiale. A cette époque l'homme était préoccupé par l'organisation des opérations militaires et surtout aériennes (nombre d'avions, la formulation à adapter, la fréquence des vols pour avoir un maximum d'efficacité,...etc.)

Après la guerre, les techniques se sont considérablement développées, grâce, notamment, à l'explosion des capacités de calcul des ordinateurs. Les domaines d'application se sont également multipliés. Les méthodes de recherche opérationnelle se sont de plus en plus appliquées aux problèmes économiques et commerciaux.

Ces méthodes se sont imposées auprès des dirigeants des grands organismes économiques et industriels comme les seuls outils permettant de prévenir aussi objectivement que possible les conséquences de leurs actions.

Une des parties essentielle de la Recherche Opérationnelle est l'optimisation combinatoire, qui s'est développée en réponse au grand nombre de problèmes pratiques qu'elle peut modéliser et qu'elle se propose de résoudre.

La plupart de ces problèmes appartiennent à la classe $\mathcal{NP}$- difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données.

Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en Recherche Opérationnelle (RO) et en intelligence artificielle (IA).

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de tailles raisonnables. Malgré les progrès réalisés (notamment en matière de la programmation linéaire en nombres entiers), comme le temps de calcul nécessaire pour

trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de tailles importantes.

Depuis une dizaine d'années, des progrès importants ont été réalisés avec l'apparition d'une nouvelle génération de méthodes approchées puissantes et générales, souvent appelées métaheuristiques. Elles sont adaptables et applicables à une large classe de problèmes.

Les métaheuristiques sont représentées essentiellement par les méthodes de voisinage comme le recuit simulé et la recherche tabou, et les algorithmes évolutifs comme les algorithmes génétiques et les stratégies d'évolution. Grâce à ces métaheuristiques, on peut proposer aujourd'hui des solutions approchées pour des problèmes d'optimisation classiques de plus grandes tailles et pour de très nombreuses applications qu'il était impossible de traiter auparavant. On constate, depuis ces dernières années, que l'intérêt porté aux métaheuristiques augmente continuellement en recherche opérationnelle et en intelligence artificielle.

Ce succès tient pour une grande part à leur capacité à fournir des solutions d'excellente qualité au prix d'une consommation en ressources réduite. La perte du caractère optimal se voit donc compensée par la diminution des temps de calcul et donc par un accroissement de la capacité de réaction. Les métaheuristiques bénéficient également d'autres avantages significatifs tels que la faculté à s'adapter rapidement à des modifications structurelles du problème (ajout ou suppression des contraintes). Ces caractéristiques les rendent parfaitement adaptées aux exigences du milieu industriel, ce qui explique l'arrivée sur le marché d'un nombre croissant d'outils d'aide à la décision intégrant des métaheuristiques [63].

Dans le monde technologique dans lequel nous vivons, la croissance des activités de distribution et de collecte de produits a créé une pression sur de nombreuses entreprises et collectivités. Ces dernières doivent résoudre des problèmes d'optimisation de grande taille liés au secteur du transport. C'est ce besoin d'optimisation dans ce secteur très concurrentiel qui a enrichi la recherche en problèmes de tournées de véhicules

Une meilleure organisation de ces dernières présente un potentiel d'économies majeur. C'est cette importance accrue des problèmes d'optimisation des tournées dans le secteur de transport qui a attiré de plus en plus les chercheurs et les gestionnaires d'entreprises pour le problème de tournées de véhicules.

Dans notre travail, nous nous intéressons à la méthode de Scatter search qui appartient à la classe des métaheuritiques. Datant de 1977 et due à Glover[25],cette approche opère sur un ensemble de solutions appelé ensemble de référence, en les combinant pour en créer de nouvelles.

Le choix de Scatter search se justifie par ses divers avantages qu'elle possède, citons :

- La possibilité de l'appliquer dans plusieurs domaines.
- Elle donne de bonnes performances à des coûts réduits.
- Cette méthode emploie des stratégies pour la diversification et l'intensification de la recherche qui ont prouvé leur efficacité dans divers problèmes d'optimisation.
- La méthodologie de Scatter Search est très flexible, puisque chacun de ses éléments peut être utilisé de différentes façons. En effet, la base de la méthode Scatter Search est représentée par l'utilisation de "cinq méthodes". Chacune des méthodes a un rôle bien déterminé.
- Utilisation d'une petite population de bonnes solutions bien diversifiées.

Puis nous nous sommes intéressés à son application au problème de tournées de véhicules.

Le premier chapitre, est une collecte de définitions de base de l'optimisation combinatoire et nous avons aussi introduit la notion de complexité et les différentes méthodes de résolution des problèmes combinatoires.

Dans le second chapitre, nous trouverons un rappel, loin d'être exhaustif mais qui permet toutefois de couvrir les tenants et aboutissants de l'essentiel de ce qui a été fait sur le problème de tournées de véhicules.

A cet effet, nous énonçons le problème, sa formulation, sa complexité qui est $\mathcal{NP}$-difficile puis nous exposons les differentes méthodes utilisées pour la résolution de ce problème, ainsi que les meilleures solutions pour quelques instances les plus connues sur les différents problèmes standards de tournées de véhicules.

Dans le troisième chapitre, nous présentons la méthode de scatter search, son historique, les différentes étapes à suivre pour l'appliquer puis une revue des travaux réalisés utilisant cette méthode.

Après avoir décrit la méthode de Scatter Search, nous avons étudié au quatrième chapitre son application au problème de tournées de véhicules, à cet effet, nous avons donné une conception générale, l'algorithme de scatter search pour ce problème et en dernier, nous avons illustrer la méthode à travers un exemple.

Chapitre 1

Optimisation Combinatoire et Métaheuristiques

1.1 Introduction

Dans ce chapitre, nous faisons de brefs rappels sur les différentes notions de base de l'optimisation combinatoire. Nous aborderons aussi quelques notions de complexité, ainsi que différentes méthodes de résolution des problèmes d'optimisation, et nous mettons l'accent sur les méthodes approchées.

1.2 Optimisation Combinatoire

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématique discrète et en informatique. Son importance se justifie d'une part, par la grande difficulté des problèmes d'optimisation et d'autre part, par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire.

Définition 1.2.1. Un problème d'optimisation combinatoire est définie par la donnée d'un ensemble fini S et d'une application $f : S \to \mathbb{R}$, il s'agit de déterminer s^* tel que $f(s^*) \leq f(s)$, pour tout élément s de S (problème de minimisation).

Définition 1.2.2. Une instance d'un problème est obtenue en spécifiant des valeurs à un ensemble significatif des parametres du problème.

Définition 1.2.3. L'ensemble de toutes les instances définit le problème.

1.3 Complexité

Une théorie de la complexité a été développée et permet de classer les problèmes faciles et difficiles.

Définition 1.3.1. A toute instance I d'un problème (P), on associe un nombre $m(I)$ qui mesure la longueur des données de cette instance et qu'on appelle la taille de l'instance I.

Définition 1.3.2. Si A est l'algorithme de résolution d'un problème (P) et I une instance de ce problème, on associe au couple (A, I) un entier $\mu(A, I)$ représentant le nombre d'opérations élémentaires (addition, multiplication, comparaison...) effectuées par l'algorithme A pour la résolution de l'instance I du problème (P). Le plus grand nombre $\mu(A, I)$ sur l'ensemble de toutes les instances ayant la même taille est appelé complexité dans le pire cas de l'algorithme A.

1.3.1 Algorithme efficace

Un algorithme est efficace si sa complexité est majorée par un polynôme en la taille des données.

1.3.2 Les classes des problèmes

Pour pouvoir exposer la notion de classes des problèmes, il est tout d'abord nécessaire de distinguer les problèmes de décision des problèmes d'optimisation combinatoire.
Un problème de décision est un problème pour lequel la réponse est « oui » ou « non ».
Notons qu'il est possible d'associer à chaque problème d'optimisation un problème de décision en introduisant un seuil k correspondant à la fonction objectif f. Le problème de décision devient : « existe-t-il une solution réalisable s telle que : $f(s) \leq \text{ou}(\geq)k$?

La classe des problèmes $\mathcal{P}$

Un problème appartient à la classe $\mathcal{P}$ s'il existe un algorithme polynomial pour le résoudre. Les problèmes appartenant à cette classe sont dits faciles [57].

La classe des problèmes $\mathcal{NP}$ et les algorithmes non déterministes

Définition 1.3.3. Un algorithme non déterministe est caractérisé par l'utilisation d'une instruction « choix » opérant sur un ensemble fini et en sélectionnant un élément sans que la manière dont le choix est effectué ne soit précisée. Ils sont caractérisés par le fait que s'il existe une manière d'effectuer le choix qui conduit à la réponse « oui » alors c'est selon cette manière que le choix est effectué.

Définition 1.3.4. Un problème de décision appartient à la classe $\mathcal{NP}$ s'il existe un algorithme non déterministe polynomial pour sa résolution.

Définition 1.3.5. Un problème de reconnaissance (ou de décision) appartient à la classe $\mathcal{NP}$ si étant donné une proposition de solution au problème, il est possible de vérifier en un temps polynomial que cette solution a pour réponse vrai.

La classe des problèmes $\mathcal{NP}$-complets

Définition 1.3.6. Un problème (P_1) se réduit polynomialement à un problème (P_2) s'il existe un algorithme polynomial de résolution de (P_1) faisant appel à la résolution de (P_2) comme procédure.

Définition 1.3.7. Un problème (P) est $\mathcal{NP}$-complet s'il est dans $\mathcal{NP}$ et si tout problème de $\mathcal{NP}$ se réduit polynomialement à (P).

Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes $\mathcal{NP}$ -difficiles et ne possèdent donc pas de solution algorithmique efficace valable pour toutes les données. Etant donné l'importance de ces problèmes, de nombreuses méthodes ont été développées en recherche opérationnelle et en intelligence artificielle. Ces méthodes peuvent être classées en deux grandes catégories : méthodes exactes et méthodes approchées.

1.4 Méthodes exactes

Les méthodes de résolution exactes sont nombreuses et se caractérisent par le fait qu'elles permettent d'obtenir une ou plusieurs solutions dont l'optimalité est garantie.

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles telles les techniques de séparation et évaluation progressive (SEP) ou les algorithmes avec retour arrière.

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de tailles raisonnables.

Comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de tailles importantes.

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille.

1.5 Méthodes approchées

Pour les problèmes de grandes tailles, les méthodes exactes ne sont pas envisageables de par leur temps de calcul qui est exponentiel et donc sans aucun intérêt en pratique. Dans ce cas, il est possible d'utiliser des méthodes approchées qui donnent des solutions réalisables, mais obtenues rapidement.

Parmi ces méthodes, on trouve les heuristiques et métaheuristiques.

1.5.1 Heuristiques

Le mot heuristique [63] vient du mot grec « heuriskêin »qui signifie trouver. En optimisation combinatoire, une heuristique est un algorithme qui fournit en temps polynomial une solution réalisable, pas nécessairement optimale pour un problème d'optimisation $\mathcal{NP}$-difficile. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch-and-Bound par exemple).

Généralement, une heuristique est conçue pour un problème particulier en s'appuyant sur sa structure propre. Lorsque les approches contiennent des principes plus généraux, on parle de Métaheuristiques.

La qualité d'une heuristique peut s'évaluer selon deux critères :

- En pratique, on implémente l'algorithme approximatif et on évalue la qualité de ces solutions par rapport aux solutions optimales (ou aux meilleures solutions connues).
- Critère mathématique : il faut démontrer que l'heuristique garantit des performances.

1.5.2 Métaheuristiques

Généralités

Le mot métaheuristique est dérivé de la composition de deux mots grecs « Méta » qui signifie au-delà et heuristique « heuriskein » qui veut dire trouver. Apparues dans les années 1980, les métaheuristiques forment un ensemble d'algorithmes visant à résoudre une large gamme de problèmes d'optimisation difficiles.

Les métaheuristiques sont une forme d'algorithmes d'optimisation stochastiques, hybridés avec une recherche locale. Le terme méta est donc pris au sens où les algorithmes peuvent regrouper plusieurs heuristiques.

Elles sont souvent inspirées par des systèmes naturels, qu'ils soient en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en éthologie (cas des algorithmes de colonies de fourmis ou de l'optimisation par essaims particulaires).

Les métaheuristiques manipulent une ou plusieurs solutions, à la recherche de l'optimum, la meilleure solution au problème. L'algorithme s'arrête après avoir atteint un critère d'arrêt, consistant généralement en l'atteinte du temps d'exécution imparti ou en une précision demandée.

Les métaheuristiques ne nécessitent pas de connaissances particulières sur le problème à optimiser pour fonctionner, le fait de pouvoir associer une (ou plusieurs) valeurs à une solution est la seule information nécessaire.

Certaines métaheuristiques sont théoriquement «convergentes» sous certaines conditions. Il est alors garanti que l'optimum global sera trouvé en un temps fini, la probabilité de se faire augmentant asymptotiquement avec le temps. Cette garanti revient à considérer que l'algorithme se comporte au pire comme une recherche aléatoire pure (la probabilité de tenter toutes les solutions tendant vers 1).

Cependant, les conditions nécessaires sont rarement vérifiées dans le cadre d'applications réelles. En pratique, la principale condition de convergence est de considérer que l'algorithme est ergodique (qu'il peut atteindre n'importe qu'elle solution à chaque mouvement), mais on se satisfait souvent d'une quasi-ergodicité (si la métaheuristique peut atteindre n'importe quelle solution en un nombre fini de mouvements).

Classification

Les métaheuristiques peuvent être classées de plusieurs façons :

On peut faire la différence entre les métaheuristiques qui s'inspirent de phénomènes naturels (colonies de fourmis, recuit simulé). Et celles qui ne s'en inspirent pas (méthode tabou), mais une telle classification ne semble pas très utile et parfois difficile de réaliser.

Elles peuvent également être classées selon leur manière d'utiliser la fonction objectif. Etant donné un problème d'optimisation consistant à minimiser une fonction objectif f sur un espace S de solutions, certaines métaheuristiques dites statiques travaillent directement sur f, alors d'autres dites dynamiques font usage d'une fonction g obtenue à partir de f en ajoutant quelques composantes qui permettent de modifier la topologie de l'espace des solutions.

Certaines métaheuristiques font usage de l'historique de la recherche au cours de l'optimisation, alors que d'autres n'ont aucune mémoire du passé. Les algorithmes sans mémoire sont en fait des processus markoviens puisque l'action à réaliser est totalement déterminée par la situation courante. Les métaheuristiques qui font usage de l'historique de recherche peuvent le faire de diverses manières.

Une autre façon de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui manipulent une seule solution.

Les méthodes qui tentent itérativement d'améliorer une solution sont appelées méthode de recherche locale ou méthode de trajectoire, dans cette optique l'algorithme fait évoluer une seule solution sur l'espace de recherche à chaque itération tandis l'autre approche manipule un ensemble de solutions en parallèle, à chaque itération.

Dans notre description, nous allons nous appuyer sur cette dernière classification et citer quelques métaheuristiques les plus utilisées.

Méthodes de recherche locale

-Recuit simulé

La méthode du recuit simulé tire son origine de la physique statistique. Son principe de fonctionnement repose sur une imitation du phénomène de recuit en science des matériaux. Le recuit est le processus physique qui consiste à porter un matériau à température élevée, puis à le refroidir d'une manière contrôlée jusqu'à l'obtention d'une structure régulière, comme dans les cristaux et l'acier. Durant ce processus, l'énergie du matériau est minimisée, en évitant de rester piégé dans certains états correspondant à des optimums locaux.

En faisant une analogie avec le processus physique décrit ci-dessus, trois chercheurs de la société IBM (S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi) ont proposé une méthode permettant d'apporter une solution aux problèmes difficiles de l'optimisation combinatoire :

Cette méthode considère une solution initiale donnée, en la modifiant, on en obtient une seconde. Soit celle-ci améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, soit celle-ci le dégrade. Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une « mauvaise » solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

-La recherche Tabou

Le principe de la méthode est : à chaque itération le voisinage de la solution courante est examiné et la solution minimisant l'augmentation du coût est sélectionnée. Pour éviter le phénomène de cyclage, la méthode interdit de revisiter une solution déjà visitée. Pour cela, une liste taboue contenant les solutions visitées est utilisée.

-Méthode de descente

Cette méthode consiste à rechercher dans le voisinage de la solution courante, une solution de coût plus faible jusqu'à arriver à un optimum local. Cette méthode a l'avantage d'être rapide mais s'arrête dés qu'un optimum local est atteint, même si celui-ci n'est pas de bonne qualité.

-Méthode GRASP

Elle a été proposée par Feo et Resende en 1995, cette méthode est une procédure itérative composée de deux phases : une phase constructive et une phase d'amélioration. Dans la première phase, on suppose qu'une solution est constituée d'un ensemble de composantes et on génère la solution pas à pas en ajoutant à chaque étape une nouvelle composante choisie dans une liste de candidats, et dans la deuxième phase, on applique une méthode de descente ou recuit simulé ou tabou.

Méthodes à population de solutions

-Les algorithmes génétiques

Les algorithmes génétiques s'inspirent de la théorie de l'évolution, initiée par Charles Darwin au XIXeme siècle.

Dans cette théorie, une population d'individus évolue grâce au mécanisme de la reproduction sexuée. Les individus les plus adaptés à leur milieu se reproduisent plus que les autres, favorisant les caractères les plus adaptés.

Au niveau de l'ADN, la recombinaison de l'ADN des deux parents (lors de la reproduction) permet de générer différentes combinaisons de gènes. Il y a des chances qu'en recombinant l'ADN de deux parents bien adaptés, l'individu généré soit encore mieux adapté.

Les algorithmes génétiques fonctionnent sur cette analogie. On part d'une population (ensemble de solutions) initiale sur laquelle des opérations de reproduction, de croisement ou de mutation vont être réalisées dans l'objectif d'exploiter au mieux les caractéristiques et propriétés de cette population puisque les bonnes solutions sont encouragées à échanger, par croisement, leurs caractéristiques et à engendrer des solutions encore meilleures.

-Optimisation par colonies de fourmis

C'est une méthode évolutive inspirée du comportement des fourmis à la recherche de nourriture. Cet algorithme a été proposé par Dorigo en 1992. Il est connu que les fourmis sont capables de déterminer le chemin le plus court entre leur nid et une source de nourriture. Ceci est possible grâce à la phéromone qui est une substance que les fourmis déposent sur le sol lorsqu'elles se déplacent. L'orsqu'une fourmi doit choisir entre deux directions, elle choisit avec une plus grande probabilité celle comportant une plus forte concentration de phéromone.

Chaque fourmi est un algorithme constructif capable de générer des solutions. Soit D l'ensemble des décisions possibles que peut prendre une fourmi pour compléter une solution partielle. La décision $d \in D$ qu'elle choisira dependra de deux facteurs, à savoir la force gloutonne et la trace.

`Force gloutonne` : est une valeur qui représente l'intérêt qu'a la fourmi à prendre la décision d, plus cette valeur est grande, plus il semble intéressant de faire le choix d.

`La trace` : représente l'intérêt historique qu'a la fourmi à prendre la décision d. Plus cette quantité est grande, plus il a été intéressant dans le passé de prendre cette décision.

On appelle optimisation par colonies de fourmis toute variation ou extension du système de fourmis décrit ci-dessus. Le premier type d'extension consiste à réaliser une recherche locale sur chaque solution produite par l'algorithme constructif, une deuxième variation consiste à prendre un certain pourcentage de décisions en ne tenant compte que de la force gloutonne, et la troisième variation consiste à faire une mise à jour supplémentaire des traces en tenant compte des choix ayant abouti à la meilleure solution.

-Optimisation par essaims particulaires

L'optimisation par essaims particulaires (OEP ou PSO en anglais) est inventée par Russel Ebenhart (ingénieur en électricité) et James Kennedy (socio psychologue) en 1995.

Cet algorithme s'inspire à l'origine du monde du vivant. Il s'appuie notamment sur un modèle développé par le biologiste Craig Raynolds à la fin des années 1980, permettant de simuler le déplacement d'un groupe d'oiseaux.

Cette méthode d'optimisation se base sur la collaboration des individus entre eux. Elle a d'ailleurs des similarités avec les algorithmes de colonies de fourmis qui s'appuient eux aussi sur le concept d'auto organisation. Cette idée veut qu'un groupe d'individus peu intelligent peut posséder une organisation globale complexe.

Ainsi, grâce à des règles de déplacement très simples (dans l'espace
des solutions), les particules peuvent converger progressivement vers un minimum local. Cette métaheuristique semble cependant mieux fonctionner pour des espaces en variables continues.

-La méthode à mémoire adaptative

La méthode à mémoire adaptative est une extension de la recherche Tabou qui permet de réaliser automatiquement une diversification et une intensification de la recherche. Cette méthode a été proposée par Rochat et Taillard en 1995. Elle fonctionne avec une mémoire centrale chargée de stocker les composantes des meilleures solutions rencontrées. Ces composantes sont combinées afin de créer de nouvelles solutions. Si la combinaison ne produit pas une solution admissible, un processus de réparation est mis en œuvre. Un algorithme de recherche locale est ensuite appliqué et les composantes de la solution ainsi obtenues sont considérées pour éventuellement faire partie de la mémoire centrale.

Chapitre 2

Le problème de tournées de véhicules

2.1 Introduction

L'une des principales préoccupations des entreprises industrielles est d'améliorer l'efficacité de leurs chaînes logistiques, pour pouvoir organiser au moindre coût un meilleur service et la fluidité de l'écoulement de leurs marchandises. Ainsi un élément fondamental de tout système logistique est la gestion et la planification des réseaux de distribution des flottes de véhicules.

L'utilisation des modèles mathématiques d'optimisation des tournées de véhicules a été l'un des succès de la recherche opérationnelle, les recherches récentes incluent des efforts significatifs en formulation du problème et en construction, analyse et implantation d'algorithmes

Les applications pratiques de ce type de problème incluent en effet : la distribution de journaux (Golden 1975)[36], le ramassage scolaire (Bennett et Gazis 1972)[6], la collecte d'ordures (Beltrami et Bodin 1974)[5], la fourniture de combustible (Garvin, Grandall, John et Spellman 1957)[21], la distribution de produits aux hypermarchés et magasins, la distribution de courrier et la gestion préventive d'inspection de routes...

Outre leur intérêt pratique, les modèles de tournées de véhicules ont été également appliqués à d'autres domaines, n'ayant apparemment aucun rapport. Des applications effectives ont été faites en ordonnancement d'atelier, où certaines situations peuvent être modélisées d'une manière identique à celle des problèmes de tournées, ainsi à titre d'exemple : le problème de recherche de séquences optimales de production sur plusieurs machines parallèles avec des coûts de lancement variables, a été résolu par A. Parker, R. Dean et R. Holmes 1977 [49] à l'aide d'un algorithme de résolution de problème de tournées. De même le problème de planification d'horaires de travail par Desrochers 1986 ou d'optimisation de réseaux de communication (B. Sanson, F. Soumis et M. Gendreau 1988)[58] ont été également résolus par des algorithmes conçus à l'origine pour la résolution des problèmes de tournées.

2.2 Le problème de tournées de véhicules et ses généralisations

Il existe un grand nombre de variantes de problèmes de tournées de véhicules (Bodin. L et Golden. B 1981)[8]. Dans une de ses formes les plus simples, on a un ensemble de n clients qui demandent un bien en quantité $q_i(i = 1, 2...n)$. Pour satisfaire ces demandes, on dispose d'un véhicule de capacité Q qui doit se réapprovisionner à partir d'un dépôt unique. Connaissant les distances entre chaque paire de clients et entre le dépôt et les clients, on cherche des tournées de longueur totale minimale telles que la quantité à livrer dans chaque tournée soit au plus égale à Q.

A partir de ce modèle de base, il est possible d'ajouter des contraintes pour arriver à des modèles plus compliqués à savoir :

- La durée de chaque tournée est limitée, et on a un temps de service pour chaque client.
- Les livraisons s'organisent en journées de travail d'une durée limitée L et toutes les commandes doivent avoir été livrées dans un laps de temps de K jours ; il est donc possible de faire plusieurs tournées par jour ; afin d'assurer l'existence d'une solution admissible, il est permis de dépasser la limite de temps L moyennant une pénalité proportionnelle à la durée du dépassement.
- On dispose d'une flotte hétérogène de véhicules ; chacun étant spécifié par sa capacité, son coût fixe et un coût variable d'utilisation dépendant de la longueur du trajet effectué.
- Le nombre de tournées est limité et on désire minimiser la longueur de la plus longue tournée.
- Les livraisons s'organisent en K journées de travail et on désire minimiser la durée de la plus longue journée de travail.
- Les clients spécifient une fenêtre de temps durant laquelle ils peuvent recevoir leurs livraisons. On désire engager un nombre minimum de véhicules pour réaliser l'ensemble des livraisons.

Le problème de tournées de véhicules a été largement étudié au cours des dernières années. Cette figure illustre les articles les plus récents concernant ce problème.

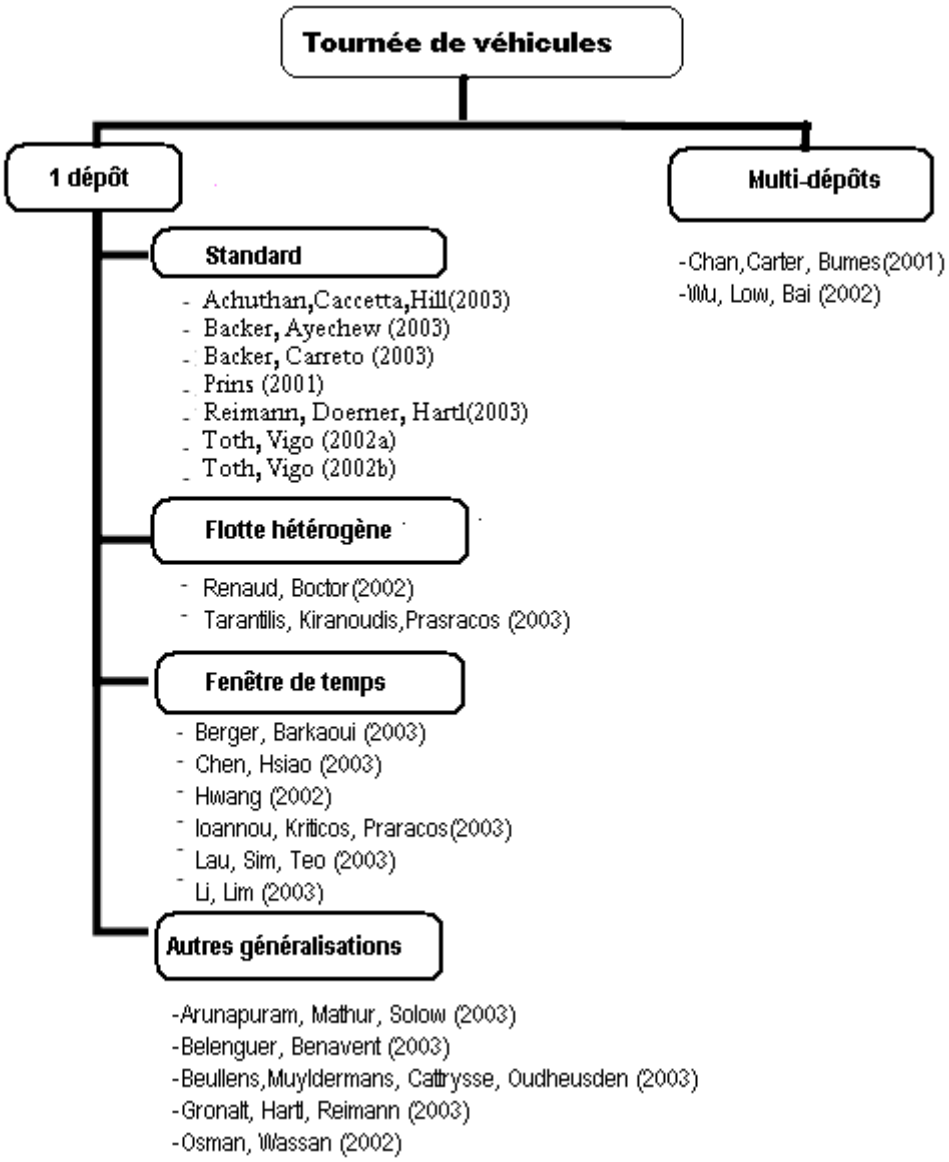

FIG. 2.1 – **Classification des problèmes de tournées de véhicules** (Source : Julie Privé (2004))

2.3 Modélisation

Le problème de tournées de véhicules est noté par VRP (Vehicle Routing Problem) qui est le nom générique le plus fréquemment utilisé dans la littérature scientifique.

Le VRP est aussi connu sous les noms de " Vehicle Scheduling " (Clark et Wright 1964 [13], Gaskel 1967) " Vehicle Dispaching " (Christofides et Eilon 1969 [11], Dantzig et Ramser 1959 [17] Gillet et Miller 1974) ou simplement " Delivery Problem " (Hays 1967).

Les modèles d'optimisation de tournées de véhicules ont également une importance théorique fondamentale.

Il existe de nombreuses formulations du problème de tournées de véhicules :
Il peut être formuler comme étant un problème de voyageur de commerce généralisé.
Plusieurs modèles de tournées de véhicules sont des variantes et extensions du problème du voyageur de commerce (TSP)(Bellmore et Nemhauser 1968).

2.3.1 Le problème de voyageur de commerce

Le TSP consiste en la détermination du parcours du coût minimal (distance, temps...etc) d'un seul véhicule partant d'une localité, visitant $n-1$ autres localités et revenant à son départ.

En théorie des graphes, le problème consiste à chercher le circuit hamiltonien de coût minimal dans un graphe complet $G = (V, A)$,valué (coût c associé à chaque arc (i, j) de A).

Nous utiliserons les notations suivantes :

$n = |V|$: Nombre de sommets du réseau.

c_{ij} : Coût de l'arc (i, j).

b_j : nombre d'arcs incident au sommet j.

a_i : nombre des arcs partant du sommet i.

$$x_i^j = \begin{cases} 1, & \text{si l'arc (i,j) appartient à la tournée ;} \\ 0, & \text{sinon.} \end{cases}$$

Une formulation fondée sur l'affectation consiste à trouver la matrice $X = x_{ij}$ de variables de décision en résolvant le problème de programmation linéaire en nombres entiers suivant :

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

$$\sum_{i=1}^{n} x_{ij} = b_j = 1, j = 1, 2, ...n \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = 1, i = 1, 2, ...n \tag{2.3}$$

$$X = (x_{ij}) \in S \tag{2.4}$$

$$x_{ij} \in \{0, 1\}, j = 1, 2, ...n \tag{2.5}$$

L'ensemble S est choisi de façon à interdire les solutions qui sont des sous tours (tours ne passant que par un sous ensemble de clients) satisfaisant les contraintes d'affectation (2.2), (2.3), (2.5). Dans la littérature, il a été proposé pour S, les ensembles suivants :

$$S = \left\{ (x_{ij}) / \sum_{i \in Q} \sum_{j \notin Q} x_{ij} > 1, \forall Q \subset V \right\} \tag{2.6}$$

$$S = \left\{ (x_{ij}) / \sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1, \forall Q \subset V \right\} \tag{2.7}$$

$$S = \{(x_{ij}) / y_i - y_j + n x_{ij} \leq n - 1, \forall y_i \in \mathbb{R}, i, j \in V, i \neq j\} \tag{2.8}$$

On peut remarquer que S contient environ 2^n contraintes d'élimination de sous tours dans les cas (2.6),(2.7), mais seulement $n^2 - 3n + 2$ contraintes dans la formulation (2.8) proposé par Miller, Tucker et Zemlin 1960. L'utilisation de l'un ou l'autre de ces ensembles dépend essentiellement de la méthode adoptée pour la résolution du problème. Algorithmiquement, les contraintes (2.7) ont été très utilisées, principalement dans les approches Lagrangiennes pour le TSP, proposé initialement par Held et Karp 1970.

Le problème du voyageur de commerce peut être interprété comme une tournée de véhicule avec un seul dépôt et un seul véhicule de capacité supérieure à la totalité de la demande. Ce modèle peut être étendu à un nombre plus grand de véhicules.

2.3.2 Le problème des m voyageurs de commerce

Le problème des m voyageurs de commerce ($m - TSP$) est une généralisation du TSP au cas où l'on désire construire m circuits hamiltoniens de coût total minimal ayant un sommet commun (le dépôt). Ce problème peut être considéré comme étant une version simplifiée du VRP où la capacité de chaque véhicule est supérieure à la totalité de la demande (ou considérée infinie) et où les m véhicules sont utilisés

En posant $a_1 = b_1 = m$, $a_i = b_i = 1$,$(i \neq 1, j \neq 1)$ dans le modèle (2.1),(2.5), on obtient la formulation du $m - TSP$, trois articles publiés indépendamment, Bellmore et al 1974 ; Orlofe 1974 et Svestka et al 1973, obtiennent des formulations équivalentes pour le $m - TSP$, dérivées de formulations pour le TSP et en conséquence prouvent que le $m-TSP$ n'est pas plus difficile que le $1-TSP$. L'équivalence s'obtient en créant m copies du point de départ, chacune connectée à tous les autres sommets exactement comme si c'était une nouvelle origine.

2.3.3 Le problème de tournées de véhicules

Le problème de tournées de véhicules (VRP) a été initialement considéré par Dantzig et Ramser 1959[17] qui ont développé une approche heuristique en utilisant des idées de programmation linéaire. Ce problème a ensuite attiré un grand nombre de chercheurs car il est théoriquement très intéressant, de plus les applications de VRP sont nombreuses. Le problème est une généralisation du $m - TSP$ au cas où chaque sommet est affecté d'un poids(demande) connu et où la somme totale des poids des sommets appartenant à un même circuit doit être bornée par une constante (capacité). De plus, on considère parfois que la somme totale des poids de ces arcs est aussi bornée par une autre constante (temps maximal par tournée).

Définition 2.3.1. En termes de la théorie des graphes, (VRP) peut être défini comme suit : Soit $G = (V, A)$ un graphe où $V = \{v_0, v_1...v_n\}$ est l'ensemble des sommets, et $A = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$ l'ensemble d'arcs. Le sommet v_0 représente le dépôt, une flotte de m véhicules identiques de capacité Q où l'ensemble de sommets $V^{'} = V - \{v_0\}$ représente n villes (ou localisation des clients), une matrice des coûts ou de distances $C = (c_{ij})$ qui satisfait l'inégalité triangulaire $c_{ij} \leq c_{ik} + c_{kj}$ est définie sur A. Quand $c_{ij} = c_{ji}$ pour tout $(v_i, v_j) \in A$, le problème devient symétrique, alors on peut remplacer l'ensemble A par un ensemble d'arêtes $E = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$, on suppose que : $m \in [\underline{m}, \overline{m}]$ avec $\underline{m} = 1$ et $\overline{m} = n - 1$, les véhicules font des collections ou des livraisons mais pas les deux à la fois.
Chaque sommet v_i lui est associé une quantité q_i $(q_0 = 0)$ d'une certaine marchandise qui doit être livrée par un véhicule
Le (VRP) consiste à déterminer un ensemble m d'itinéraires de capacité minimale commençant par v_0 et se terminant en v_0. Chaque sommet $v_i \in V^{'}$ est visité une seule fois par un seul véhicule, la quantité totale attribuée à chaque itinéraire ne dépasse pas la capacité Q du véhicule qui assure le service de la route.

On définit une solution du VRP comme étant un ensemble de m routes,
$S = \{R_1, R_2..R_m\}$,$R_K = \{v_0, v_{R_1}, v_{R_2}, ..v_0\}$, où R_k représente un ensemble ordonné de sommets consécutifs représentants l'itinéraire k par conséquent on note $v_i \in R_k$ si v_i est une composante de R_k et de la même manière si $(v_i, v_i) \in R_k$ si v_i,v_j sont deux sommets consécutifs dans R_k. Enfin le coût d'une solution S est défini comme :
$C(S) = \sum_{1 \leq k \leq m} \sum_{(i,j) \in R_k} C_{ij}$

Définissons maintenant l'ensemble des variables nécessaires pour effectuer la formulation mathématique.

Paramètre

m : nombre de véhicules disponibles.

n : nombre de client à visiter. Les clients sont numérotés de 1 à n, et l'entrepôt a le numéro 0.

Q_k : capacité du véhicule k.

q_i :la demande du client i.

c_{ij} : coût ou distance entre la ville i et j

Variables de décision

y_{ik} : variable de décision binaire qui est égale à 1 si la commande du client i est livrée par le véhicule k, 0 autrement.

x_{ijk} : variable de décision binaire qui est égale à 1 si le véhicule k voyage de la ville i vers la ville j et à 0 autrement.

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} \sum_{k=1}^{m} x_{ij}^{k} \tag{2.9}$$

$$\sum_{i=1}^{n} q_i y_i^k \leq Q_k, k = 1, 2...m \tag{2.10}$$

$$\sum_{k=1}^{m} y_i^k = \begin{cases} m, & \text{i=0 ;} \\ 1, & \text{i=1,..m.} \end{cases} \tag{2.11}$$

$$\sum_{i=0}^{n} x_{ij}^k = y_i^k, j = 0, ...n, k = 1, ..m \tag{2.12}$$

$$\sum_{i,j \in U}^{n} x_{ij}^k \leq |U| - 1, 1 \leq |U| \leq n - 1, i = 0, ...n, k = 1, ..m \tag{2.13}$$

$$y_i^k, x_{ij}^k \in \{0, 1\}, i, j = 0, ...n, k = 1, ...m \tag{2.14}$$

Dans cette formulation (Fisher et Jaikumar 1981)[20] les variables x_{ij}^k indiquent si (v_i, v_j) est une arête de l'itinéraire k dans la solution optimale, et de même y_i^k spécifie si l'itinéraire k contient le sommet v_i.

Contrainte(2.10) : garantit que la capacité du véhicule n'est pas dépassée.

Contrainte(2.11) : assure que chaque client est visité par exactement un seul véhicule et m véhicules visitent le dépôt v_0.

Contrainte(2.12) : assure que chaque client est visité par le même véhicule qui le quitte.

Contrainte(2.13) : Sous tournée à éliminer.

Contrainte(2.14) : définit la contrainte d'intégralité.

Cette formulation contient deux problèmes bien connus : le problème d'affectation généralisé (PAG) spécifié par les contraintes (2.1), (2.2) et (2.5) et le problème de voyageur de commerce (TSP) résulte quand y_{ik} sont fixés pour satisfaire les contraintes PAG pour un k. Plus précisément, PAG est un sous problème responsable de l'attribution des clients pour les routes des véhicules et l'ordre dans lequel les clients sont visités par un véhicule est déterminé par la solution du TSP sur chaque itinéraire.

2.4 Complexité du VRP

Le VRP a été prouvé par Lenstra et Rinnooy 1981 [46], comme étant NP-difficiles, c'est à dire qu'il n'existe aucun algorithme en temps polynomial pour le résoudre.
La preuve consiste au fait que le VRP peut être réduit à un problème de TSP qui se réduit également au problème de recherche d'un circuit hamiltonien qui est un problème NP-difficile.

2.5 Les méthodes de résolution

Le VRP appartient au noyau des problèmes de gestion de distribution et a été intensivement étudié. (Christofides, Mingozzi et Toth 1979 [12], Bodin, Golden, Assad et Ball 1983[9], Christofides 1985, Laporte et Nobert 1987 [44] Golden et Assad 1988 [37])

La recherche des méthodes efficaces de résolution des problèmes de tournées de véhicules, a été à l'origine d'importants développements en Programmation Mathématique et en Optimisation Combinatoire, par la mise au point, l'analyse et l'implantation d'algorithmes et d'heuristiques plus performants pour résoudre des problèmes plus complexes.

Les techniques qui ont été mises en œuvre pour résoudre ces problèmes sont de deux types :

- Celles qui résolvent le problème jusqu'à l'optimalité fondées sur des techniques de partitionnement et d'évaluation successives (Branch and Bound).

- Celles qui donnent des solutions approchées.

2.5.1 Méthodes exactes

Comme le VRP est $\mathcal{NP}$-difficiles, les algorithmes existants permettent de résoudre des problèmes comportant un peu plus de 25 sommets, mais dés que le problème devient plus grand, les temps de calcul deviennent prohibitifs. En fait, aucune de ces méthodes n'a été mise en pratique.

D'après Laporte et Nobert 1987[44] les méthodes exactes pour résoudre le VRP sont généralement de trois types :

1. Les méthodes de recherche arborescente (Branch and Bound).
2. Génération de coupe.
3. La programmation dynamique.

Les méthodes de recherche arborescente

Développé par Laporte, Mercure et Nobert 1986 exploitent les ressemblances entre le VRP et $mTSP$. Avella, Boccia et Sforza (2003) ont utilisé cette approche pour résoudre le problème réel de livraison de carburant. Shutler (2001), présente une amélioration à un algorithme de séparation et évaluation progressive afin de résoudre le problème de TSP symétrique. L'auteur définit une nouvelle règle de séparation qui permet de réduire le nombre de décisions requises à chacun des nœuds pour trouver une solution au problème. Les problèmes résolus sont de 100 à 500 nœuds. Toth et Vigo (2002a) abordent le problème de tournées de véhicules avec restriction de capacité. La taille des problèmes résolus a grandement évolué, Toth et Vigo (2002a) ont réussi à faire passer le nombre de clients, pour lequel il était possible de résoudre le problème de tournées de véhicules avec contraintes de capacité, de 25 à 100 clients, ce qui représente un progrès considérable.

Génération de coupe

La génération de coupe, en anglais "branch and cut" est une généralisation de la séparation et l'évaluation progressive dans lequel plusieurs coupes sont générées afin de restreindre l'espace des solutions et ainsi augmenter la valeur de la relaxation linéaire.

Cabral, Gendreau, Ghiani et Laporte (2003) présentent une solution pour le " hierarchical Chinese postman problem". Ils réussissent à résoudre des problèmes avec 150 nœuds dans le cas où l'objectif consiste à minimiser le temps de route et de 50 nœuds lorsqu'il s'agit d'un objectif hiérarchique. Belenguer et Benavent (2003) ont traité aussi le problème où les demandes sont sur les arcs, par contre, ils considèrent plusieurs véhicules et non un seul. Dans leurs cas, des problèmes de 50 nœuds et 97 arcs pour lesquels un service est requis ont été solutionnés. L'article de Achuthan, Caccetta et Hill (2003) aborde le problème de tournées de véhicule classique. Les auteurs développent une nouvelle génération de coupes et une procédure de recherche permettant d'identifier les contraintes qui sont violées. Les tests ont été effectués sur 1650 problèmes de 15 à 100 clients.

Programmation dynamique

Il est parfaitement connu que seul un très petit nombre de problèmes d'optimisation combinatoire peuvent être résolus par la programmation dynamique. L'explosion de nombre d'états rend l'utilisation de cette technique impossible suite aux temps de calculs prohibitifs.

La programmation dynamique pour le problème de VRP a été initialement proposée par Eilon, Watson-Gandy et Christofides 1971 et en 1981, dans l'article de Christofides, Minggozzi et Toth, cette méthode a été utilisé pour les problèmes dont le nombre de sommets est $10 \leq n \leq 25$.

Les algorithmes exacts pour résoudre les problèmes $\mathcal{NP}$-difficiles nécessitent un

nombre de calcul qui augmente exponentiellement avec la taille du problème.

A cause du succès limité des méthodes exactes beaucoup d'attention a été portée dans la littérature au développement d'algorithmes heuristiques.

2.5.2 Méthodes approchées

La plupart des travaux réalisés sur le VRP sont liés à des méthodes approchées se contentent de trouver des solutions aussi bonnes que possible en temps raisonnable sans garantir l'optimalité.

Les heuristiques qui sont été développées pour le VRP sont fondées sur des heuristiques initialement développées pour résoudre le TSP.

Les méthodes approchées peuvent être classées en quatre grandes familles :

1. Méthodes constructives
2. Méthodes en deux phases.
3. Méthodes d'optimisation incomplète
4. Méthodes d'amélioration

Algorithmes pour le VRP standard

Quatre articles récents traitent le problème de tournées de véhicules à un seul dépôt avec contrainte de capacité. Dans ce cas, un ensemble de clients avec une demande et une adresse connues requièrent une livraison à partir d'un dépôt unique et avec une flotte de véhicules identiques. La demande cumulée des clients se trouvant sur la même route ne doit pas dépasser la capacité des véhicules utilisés. L'objectif consiste à minimiser la distance parcourue par tous les véhicules pour visiter une et une seule fois tous les clients puisque ce problème ne tient pas compte de la limite sur la durée des tournées, il est nommé le " capacitated vehicle routing problem " (CVRP).

Toth et Vigo(2002a) présentent une revue des algorithmes exacts utilisés pour résoudre le CVRP. Dans leur papier, seulement les restrictions sur la capacité des véhicules sont imposées et l'objectif consiste à minimiser le coût total (i.e le nombre de route et/ou leur longueur ou le temps de route nécessaire pour servir tous les clients). Par conséquent, il n'y a aucune contrainte concernant la durée maximale des routes. On considère deux cas, soit celui où les distances sont symétriques et le second où elles sont asymétriques. Les auteurs présentent deux algorithmes de " branch and bound " pour résoudre ce problème. Les algorithmes exacts permettent de résoudre des problèmes CVRP asymétriques allant jusqu'à 300 points et quatre véhicules. Ces résultats sont obtenus en 1000 CPU secondes. Dans leur revue, Toth et Vigo (2002a) discutent des algorithmes exacts permettant de résoudre le CVRP symétrique. Ils présentent les algorithmes de Fisher (1994) et Miller (1995) en soulignant qu'il est difficile de comparer entre eux les algorithmes puisque chacun des auteurs posent des hypothèses différentes

sur les mêmes problèmes. Par exemple, Fisher (1994)interdit les routes qui visitent un seul client tandis que Miller (1995) les permet.
Pour résoudre le CVRP, Baker et Ayechew (2003) utilisent un algorithme génétique. Les distances obtenues par l'algorithme génétique se situent en moyenne à 0.5% de l'optimum.

Ces tests ont été effectués sur des problèmes de 50 à 199 points dont les résultats optimaux de la littérature sont présentés au tableau (2.1). Ils trouvent la solution optimale à quatre reprises. L'article de Achuthan, Caccetta et Hill (2003) traite lui aussi du CVRP à l'aide de la méthode génération de coupe. Plus spécifiquement, ils développent de nouvelles coupes et une procédure de recherche permettant d'identifier les violations comme par exemple, la création de sous-tour. Les tests ont tout d'abord été effectués sur 1 650 problèmes générés aléatoirement. Par la suite, 24 problèmes standard de la littérature ont été testés. Pour chacun des problèmes, le nombre de véhicules est fixé au nombre minimal de camions requis. Le temps maximal imposé pour trouver une solution est d'une heure, c'est pourquoi dans le tableau(2.1), quelques bornes supérieures ne sont pas disponibles (N/A). Le nombre de clients est de 44 à 199.

Le tableau (2.1) présente les résultats de 14 problèmes présentés par Achtuhan, Caccetta et Hill. Lorsque les routes à un seul client sont permises, les problèmes 11, 18 et 19 ont été résolus optimalement en respectivement 487.4, 35.3 et 172.2 secondes. Pour le problème 17, une meilleure solution a été trouvée mais il n'est pas prouvé que le résultat est optimal puisque la limite de temps de calcul a été atteinte. Les temps de résolution pour les problèmes 11, 18, 19 et 17 sont de 488.7, 15.4, 176.3 et 458 secondes respectivement.

N^0 du problème	Nb de client	Route à un seul client permises		Route à un seul client interdit		Meilleurs solution connue	Fisher(1994)		Publication
		Borne inf	Borne sup	Borne inf	Borne sup		Borne inf	Borne sup	
11	50	514,58	524,61	514,58	524,61	$524,61^*$	507,09	$\frac{N}{A}$	Chritofides, Eilon(1969)
12	75	786,29	$\frac{N}{A}$	786,29	$\frac{N}{A}$	835,26	755,05	$\frac{N}{A}$	Chritofides, Eilon(1969)
13	100	799,07	$\frac{N}{A}$	799,07	$\frac{N}{A}$	826,14	785,86	$\frac{N}{A}$	Christifides, Eilon(1969)
14	150	950,65	$\frac{N}{A}$	950,57	$\frac{N}{A}$	1028,14	932,68	$\frac{N}{A}$	Chritofides, et al(1979)
15	199	1114,17	$\frac{N}{A}$	1153,99	$\frac{N}{A}$	1298,79	1096,72	$\frac{N}{A}$	Chritofides et al(1979)
16	120	862,98	$\frac{N}{A}$	862,98	$\frac{N}{A}$	1042,11	$\frac{N}{A}$	$\frac{N}{A}$	Christofides et al(1979)
17	100	785,29	819,56	813,73	819,56	$819,56^*$	817,77	819,56	Chritofides et al(1979)
18	44	721,88	723,54	723,54	723,54	$723,54^*$	720,76	723,54	Fisher (1994)
19	71	238,51	241,97	238,51	241,97	$241,97^*$	237,76	241,97	Fisher (1994)
20	134	1120,16	$\frac{N}{A}$	1120,16	$\frac{N}{A}$	1163,60	1133,73	1163,60	Fisher (1994)
21	75	659,17	752,25	659,17	752,25	$\frac{N}{A}$	$\frac{N}{A}$	$\frac{N}{A}$	Reinelt (1981)
22	75	705,25	949,26	705,25	949,26	$\frac{N}{A}$	$\frac{N}{A}$	$\frac{N}{A}$	Reinelt (1981)
23	75	932,41	$\frac{N}{A}$	932,41	$\frac{N}{A}$	$\frac{N}{A}$	$\frac{N}{A}$	$\frac{N}{A}$	Reinelt (1981)
24	100	1000,4	$\frac{N}{A}$	1003,17	1175,19	$\frac{N}{A}$	$\frac{N}{A}$	$\frac{N}{A}$	Reinelt (1981)

TAB. 2.1 – **Résultats du CVRP.**(Source : Achuthan, Cacetta et Hill(2003))

Étant donné que les algorithmes exacts ne réussissent toujours pas à trouver des solutions pour des problèmes de grandes tailles, plusieurs chercheurs se sont penchés vers des heuristiques. D'énormes progrès ont été effectués concernant les problèmes de tournées de véhicules. Taillard (1993)[59] a apporté une grande contribution en trouvant la meilleure solution pour la plupart des problèmes de la littérature. Son heuristique consiste essentiellement à décomposer le problème en sous-problèmes et il propose deux méthodes distinctes de partitionnement. Chacun des sous-problèmes est résolu en utilisant une approche tabou. Après un certain nombre d'itérations, les sous-problèmes sont réunis pour reformer le problème entier. Le tableau (2.2), présente une comparaison de la performance des métaheuristiques utilisées au cours des dix dernières années. Les heuristiques sont classées selon le pourcentage d'écart moyen par rapport à la meilleure solution connue. On peut y remarquer que le type d'algorithme le plus utilisé pour résoudre le problème de tournée de véhicule est celui de la recherche tabou. Le tableau (2.3) présente les meilleures solutions connues sur les dix problèmes classiques.

Algorithme utilisé	% écart par rapport à la meilleure solution	Nb de meilleures solutions trouvées	Temps moyen (min)	Publication
Tabou	0,05	12	-	Taillard (1993)
Algorithme génétique	0,08	10	5,2	Prins(2001)
Tabou	0,1	5	103	Xu-Kelly (1996)
Tabou	0,2	9	-	Gedreau et al (1994)
Tabou	0,39	6	-	Taillard(1993)
Tabou	0,55	6	-	Rego et Rocairol(1996)
Tabou granulaire	0,55	4	3,5	Toth et Vigo(2002b)
Tabou	0,68	5	-	Gendreau et al(1991)
Tabou	0,77	4	-	Rego et Rocairol(1996)
Tabou	0,86	5	46,8	Gendreau et al(1994)
Algorithme génétique (3000 croisements max)	0,9	5	0,7	Prins(2001)
Tabou	1,01	4	26,1	Osman(1993)
Tabou	1,03	3	34	Osman(2003)
Recuit simulé	2,09	2	-	Osman(1993)

TAB. 2.2 – **Comparaison des métaheuristiques utilisées.**(Source :Prins(2003))

Nom	Nb clients	Capacité véhicules	Longueur max route	Temps de service	Meilleure solution	Publication
C1	50	160	∞	0	524,61	Taillard(1993)
C2	75	140	∞	0	835,26	Taillard(1993)
C3	100	200	∞	0	836,14	Taillard(1993)
C4	150	200	∞	0	1028,42	Taillard(1993)
C5	199	200	∞	0	1291,45	Rochat et Taillard (1995)
C6	50	160	200	10	555,43	Taillard(1993)
C7	75	140	160	10	909,68	Taillard(1993)
C8	100	200	230	10	865,94	Taillard(1993)
C9	150	200	200	10	1162,55	Taillard(1993)
C10	199	200	200	10	1395,85	Rochat et Taillard (1995)

TAB. 2.3 – **Résultats sur les problèmes de Christofides et Eilon(1989).**(Source : Reimann, Doerner, Hart(2003))

Deux autres articles présentent aussi des résultats très intéressants pour le problème de VRP à un seul dépôt, il s'agit de Prins (2001)[50] et Toth et Vigo (2002b). Prins (2001) privilégie l'algorithme génétique tandis que Toth et Vigo (2002b) adoptent la recherche taboue. Ces derniers proposent dans leur algorithme une réduction de l'espace de recherche.

D'autre part, Baker et Carreto (2003) présentent une interface qui a été développée afin de résoudre les problèmes de VRP. Ils ont utilisé une heuristique basée sur une recherche adaptative aléatoire gloutonne. Le système décrit permet de combiner les connaissances et l'intuition avec la puissance de l'ordinateur afin de trouver de bonnes solutions. Ainsi, l'utilisateur a un contrôle sur les routes finales ce qui est beaucoup plus acceptable en pratique. Une séance interactive d'au plus 15 minutes permet de trouver des routes intéressantes. Les tests effectués sur des problèmes de 50 à 199 points permettent d'obtenir des résultats à moins de 1% de la meilleure solution connue.

Pour les problèmes de grande taille entre 240 et 483 clients, ce sont Reimann, Doerner et Hartl (2003) qui obtiennent les meilleurs résultats. Ces derniers trouvent une meilleure solution pour 10 des 20 problèmes. De plus, ils égalisent quatre autres résultats dont la meilleure solution a déjà été trouvée. L'heuristique utilisée pour trouver ces solutions est une heuristique composite inspirée de l'algorithme d'optimisation par colonie de fourmis. Effectivement, Reimann, Doerner et Hartl ont utilisé le principe des phéromones lors de la construction de leur heuristique. Par contre, d'autres algorithmes ont aussi été utilisés, par exemple, on retrouve la 2-opt, l'algorithme de balayage de Gillet et Miller (1974) et l'algorithme modifié de Miehle présenté par Spaeth (1977). Ces résultats sont résumés au tableau (2.4)

Nom	Nb clients	Capacité véhicules	Longueur max route	Temps de service	Meilleure solution	Publication
1	240	550	650	0	5644,02	Reimann, Doerner, Harti(2003)
2	320	700	900	0	8447,92	Prins(2001)
3	400	900	1200	0	11036,22	Prins(2001)
4	480	1000	1600	0	13624,52	Prins(2001)
5	200	900	1800	0	6460,98	Prins(2001)
6	280	900	1500	0	8412,80	Prins(2001)
7	360	900	1300	0	10195,59	Prins(2001)
8	440	900	1200	0	11828,78	Prins(2001)
9	255	1000	∞	0	586,87	Reimann, Doerner, Harti (2003)
10	323	1000	∞	0	746,56	Golden et al(1998)
11	399	1000	∞	0	927,27	Reimann, Doerner, Harti (2003)
12	483	1000	∞	0	1133,79	Reimann, Doerner, Harti (2003)
13	252	1000	∞	0	865,07	Reimann, Doerner, Harti (2003)
14	320	1000	∞	0	1086,24	Reimann, Doerner, Harti (2003)
15	396	1000	∞	0	1358,21	Reimann, Doerner, Harti (2003)
16	480	1000	∞	0	1635,16	Reimann, Doerner, Harti (2003)
17	240	1000	∞	0	708,76	Reimann, Doerner, Harti (2003)
18	300	200	∞	0	998,83	Reimann, Doerner, Harti (2003)
19	360	200	∞	0	1367,20	Reimann, Doerner, Harti (2003)
20	420	200	∞	0	1822,94	Reimann, Doerner, Harti (2003)

TAB. 2.4 – **Problèmes de grandes tailles.**(Source : Reimann, Doerner, Hart(2003))

Chapitre 3

Scatter Search

3.1 Définition et origine

La recherche par dispersion ou scatter search est une méthode évolutive ancienne décrite par Glover en 1977 [25] mais cette technique est restée dans sa conception théorique et n'a connu de succès que vers les années quatre-vingt dix où les études ont montré les avantages pratiques de cette méthode pour résoudre divers problèmes d'optimisation.
La recherche par dispersion opère sur un ensemble de solutions appelé ensemble de référence, en les combinant pour en créer des nouvelles.

L'approche provient de recherche de stratégies pour la création de règle de décision et de contraintes de substitution.

Historiquement, les stratégies pour combiner des règles de décision ont été introduites dans le contexte des méthodes de programmation pour obtenir une règle de décision locale améliorée pour les problèmes d'ordonnancement d'ateliers. De nouvelles règles ont été générées numériquement en créant des combinaisons pondérées des règles existantes.

L'approche a été motivée par la supposition que des informations sur le rapport souhaitable de choix alternatif sont capturées sous différentes formes par des règles différentes, et que ces informations peuvent être exploitées de manière plus efficace lorsqu'elles sont intégrées au moyen d'un mécanisme de combinaison que lorsqu'elles sont traitées par la norme de stratégies des règles différentes en sélectionnant une à une de façon isolée.

Les règles de décision créées à partir de cette combinaison produisent de meilleurs résultats empiriques.

Les procédures pour combiner les contraintes emploient un mécanisme de génération des combinaisons pondérées, dans ce cas appliqué dans le cadre de la programmation en nombres entiers non linéaire pour créer des nouvelles appelées contraintes de substitution[25].

L'approche isole des sous ensembles de contraintes qui ont été évalués à être plus critiques par rapport à des solutions basées sur les contraintes de substitution et produit des nouvelles pondérations qui reflètent la mesure dans laquelle la composante était satisfaite ou violée .

Le principal rôle des contraintes de substitution est de fournir des moyens d'évaluer les choix qui pourraient être utilisés pour générer et modifier des solutions d'essai. De cette fondation, une variété de procédés heuristiques évolués qui font usage de leurs contraintes de substitution.

En conséquence, ces processus ont abouti à la stratégie complémentaire de la combinaison des solutions et des contraintes qui sont devenues manifestes dans la recherche dispersée.

3.1.1 Principes de Scatter search

Scatter search est fondée sur trois principes :

1. Les informations pertinentes de l'emplacement de la solution optimale sont généralement contenues dans un ensemble diversifié de l'ensemble de solutions « élite »(générées).
2. Grâce à une bonne combinaison de solutions, il est possible de réaliser d'autres régions intéressantes qui pourraient contenir la meilleure solution ou les solutions qui peuvent conduire à la solution optimale.
3. Tenir compte de multiples solutions comme fondation d'en créer de nouvelles, Scatter search favorise l'exploration de l'information ne figurant pas dans chaque solution individuellement.

3.2 Procédure de Scatter search

La recherche par dispersion comprend cinq étapes :

1. Une méthode de génération de diversification pour produire une collection de solutions diverses, en utilisant une solution d'essai arbitraire comme entrée.
2. Une méthode d'amélioration pour transformer une solution initiale en une ou plusieurs solutions d'essai améliorées.
3. Une méthode de mise à jour de l'ensemble de référence pour construire et maintenir un ensemble de référence comprenant les meilleures solutions trouvées. L'incorporation de solutions à l'ensemble de référence est effectuée selon leur qualité ou leur diversité.
4. Une méthode de génération d'un sous ensemble pour opérer sur l'ensemble de référence pour produire un sous ensemble de ces solutions comme base pour créer des solutions combinées.
5. Une méthode de combinaison de solutions pour transformer un sous ensemble donné de solutions produites par la méthode de génération d'un sous ensemble en un ou plusieurs vecteurs combinés de solutions.

La population initiale P

La population initiale est générée en utilisant des stratégies de diversification et d'intensification.

La diversification (ou exploration)

C'est une stratégie qui sert à conduire la recherche vers des nouvelles régions afin d'avoir un échantillon représentatif de l'espace de recherche dans le but d'atteindre l'optimum global.

L'intensification (ou exploitation)

Vise à utiliser l'information déjà récoltée pour définir et parcourir les zones intéressantes de l'espace de recherche pour atteindre la meilleure solution.

Le générateur de diversification est utilisé pour commencer à générer l'ensemble P de solutions initiales de taille Psize. Le cardinal de l'ensemble P est généralement égal au $max(100, 5 * b)$ où b est le cardinal de l'ensemble de référence.

Des stratégies de diversification sont utilisées pour avoir une population variée qui permettra d'explorer des régions diverses de l'espace de recherche.

A chaque solution S, un coût est associé dans le but d'établir une liste de solutions ordonnées selon la valeur de la fonction objectif.

Pour avoir une population de solutions qui représente des zones diverses de l'espace de recherche, on utilise une méthode de diversification.

3.2.1 Méthode de génération de diversification

Nous indiquons deux types de diversification, l'un appliqué aux problèmes d'optimisation en 0–1 et l'autre pour les problèmes qui peuvent être formulés comme une optimisation d'éléments d'une permutation.

a-Génération de vecteurs en zéro - un

Soit x un n-vecteurs de composantes x_j, $j = 1, 2.....n$ de valeur 0 ou 1 où n est le nombre de variables du problème considéré. Soit h un paramètre tel que $h \leq n - 1$

Soit $x = (0, 0, 0...0)$ la solution initiale et pour chaque valeur de $h = 1, 2, \ldots .n - 1$, on génère deux types de solutions $x^{'}$ et $x^{''}$ de la manière suivante :

Pour type1

$$x'_1 = 1 - x_1$$
$$x'_{1+hk} = 1 - x_{1+hk} \text{ pour } k = 1, 2, \ldots, \frac{n}{h}$$

Pour type2

$x^{''}$ est le complémentaire de $x^{'}$

b-Générateur de diversification pour les problèmes de permutation

Tous les problèmes de permutation peuvent être formulés comme des problèmes en $0-1$, ils constituent une classe spéciale qui doit être traitée différemment.

On considère la permutation p de n éléments donnée dans l'ordre consécutif comme une solution initiale $p = (1, 2, 3 \ldots..n)$, on définit $p(h : s) = (s, s + h, +2h, s + 3h, \ldots..s + rh)$ où s est un entier positif entre 1 et h, r est le plus grand entier positif tel que $s + rh \leq n$.
Et on définit la permutation
$p(h) = (p(h : h), p(h : h - 1), p(h : h - 2) \ldots\ldots, p(h : 1))$

3.2.2 Ensemble de solutions de référence

L'ensemble de référence, $Refset$, est une collection de solutions de qualité et de solutions diverses qui sont employées pour produire de nouvelles solutions par l'application de la méthode de combinaison de solution. L'ensemble de référence comprend l'union de deux sous-ensembles, $Refset_1$ et $Refset_2$, de taille b_1 et b_2 respectivement. C'est-à-dire, $|Refset| = b = b_1 + b_2$. La construction de l'ensemble de référence initial commence par le choix des meilleures solutions b_1 à partir de P. Ces solutions sont ajoutées à $Refset$ et supprimées de P.

Pour chaque solution x améliorée dans $P - Refset$, le minimum des distances aux solutions y dans $Refset$ est calculé comme suit :

$$d_{min}(x) = \min_{y \in refset} \{d(x, y)\}$$

On définit une distance entre deux solutions x et y de la manière suivante :

$$d(x, y) = \sum_i |x_i - y_i|$$

Puis, la solution avec le maximum de ces distances minimales est choisie. Cette solution est ajoutée à $Refset$ et supprimée de P. On répète le processus jusqu'à avoir b_2 solutions qui formeront l'ensemble $refset_2$
L'ensemble de référence résultant a les solutions b_1 de haute qualité et les solutions b_2 diverses.
Le nombre de solutions contenues dans l'ensemble de référence doit être ≤ 20.

3.2.3 Méthode de génération des sous ensembles

Cette méthode consiste à générer les sous ensembles qui seront utilisés pour créer la structure des combinaisons de l'étape suivante en utilisant les éléments de $Refset$ construits dans l'étape précédente.

On distingue quatre types de sous ensembles

Type1 : Combiner les solutions de l'ensemble de référence $Refset$ deux à deux.

Type2 : Combiner les solutions de $Refset$ trois à trois en prenant chaque sous ensemble de type 1 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type3 : Combiner les solutions de $Refset$ quatre à quatre en prenant chaque combinaison de type 2 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type4 : Combiner les i meilleures solutions de $Refset$ au sens de la fonction objectif avec i variant de 5 à b

Noter que le nombre de sous ensembles de type 1 produits égal à $\frac{b^2-b}{2}$

Si l'ensemble de référence contient b solutions, la procédure examine approximativement $\frac{(3b-7)*b}{2}$ combinaisons de quatre types cités précédemment.

3.2.4 Méthode de combinaison des solutions

Cette méthode emploie les sous-ensembles produits avec la méthode de génération pour combiner les éléments dans chaque sous-ensemble en vue de créer de nouvelles solutions. Généralement, la méthode de combinaison de solution est un mécanisme spécifique du problème, puisqu'on le lie directement à la représentation de solution. Selon la forme spécifique de la méthode de combinaison de solution, chaque sous-ensemble peut créer une ou plusieurs nouvelles solutions.

Considérons la méthode suivante de combinaison pour les solutions en 0-1 .

On définit un score pour chaque variable en se basant sur les solutions du sous ensemble et la valeur de la fonction objectif correspondante comme suit : Score de la variable i qui correspond à la solution d'un sous ensemble E est :

$$score(i) = \frac{\sum_{j \in E} vobj(j) * x_i^j}{\sum_{j \in E} vobj(j)}$$

$vobj(j)$: valeur de la fonction objectif donnée par la solution j
x_i^j : valeur de la fonction objectif de la solution j

La solution sera construite de la manière suivante :

$$x_i^j = \begin{cases} 1, & \text{si le score } (i) > 0.5 \\ 0, & \text{si le score } (i) \leq 0.5 \end{cases}$$

Une fois les solutions sont calculées, on peut obtenir celles qui ne sont pas réalisables, à l'aide d'une méthode d'amélioration appliquée à chacune des solutions, on obtient une ou plusieurs solutions de chaque type.

Si une solution x n'est pas dans $Refset$ et que la valeur de la fonction objectif de x est meilleure que la valeur de la fonction objectif du plus mauvais élément de $Refset_1$ alors ajouter x à $Refset_1$ et supprimer le mauvais élément en cours dans $Refset_1$ (le mauvais élément est celui qui a la mauvaise valeur de la fonction objectif).

Si x n'est pas dans $Refset_2$ et $d_{min}(x)$ est supérieur à $d_{min}(y)$ pour une solution dans $Refset_2$ alors :

Ajouter à $Refset_2$ et supprimer le mauvais élément en cours dans $Refset_2$. (le mauvais élément est la solution y avec la valeur la plus petite de $d_{min}(y)$.

3.3 Algorithme de Scatter search

Pour illustrer les différentes phases de la méthode de scatter search, on définit les paramètres suivants :

$Psize$: la taille de l'ensemble de solutions générées par la méthode de génération de diversification.

b : cardinal de l'ensemble de référence.

b_1 : cardinal du sous ensemble de référence contenant les meilleures solutions au sens de la fonction objectif.

b_2 : cardinal du sous ensemble de référence contenant les solutions diverses.

$Maxlter$: nombre maximum d'itérations.

$Refset$: ensemble de référence.

$Refset_1$: sous ensemble contenant les meilleures solutions.

$Refset_2$: sous ensemble contenant les solutions diverses.

Algorithm 1 Scatter Search

1. Commencer par $P = \emptyset$. Utiliser une méthode de génération de diversification pour construire une solution x. Appliquer une méthode d'amélioration à x pour obtenir une solution améliorée x^*.
   Si $x^* \notin P$, $P = P \bigcup x^*$, sinon omettre x^*. Répéter jusqu'à $|P| = PSize$.

2. Ordonner les solutions de P par rapport à la fonction objectif de la meilleure à la mauvaise solution.

   Pour(iter=1 **à** Maxlter)

3. Etablir $Refset = Refset_1 \bigcup Refset_2$ de P avec $|Refset| = b$, $|Refset_1| = b_1$, $|Refset_2| = b_2$.

   Prendre les premières solutions b_1 dans P et les ajouter à $Refset_1$. Pour chaque solution dans $P - Refset$ et $y \in Refset$, calculer une mesure de distance de dissimilitude $d(x, y)$
   solution dans P- Refset et y Refset, calculer une mesure de distance de dissimilitude .
   Choisir la solution $x^{'}$ qui maximise $d_{min}(x) = \min\limits_{y \in refset} \{d(x, y)\}$, ajouter $x^{'}$ à $Refset_2$, jusqu'à $Refset_2 = b_2$.
   Faire Nouveaux Eléments = VRAI.

   Tant que (Nouveaux Eléments) **Faire**

4. Calculer le nombre de sous ensembles (MaxSubset) qui incluent au moins un nouvel élément.
   Faire Nouveaux Eléments = FAUX.
   Pour(Subset =1,...., MaxSubset) **Faire**

5. Générer le sous ensemble s suivant avec la méthode de génération de sous ensemble. Cette méthode génère l'un des quatre types des sous ensembles avec le nombre d'éléments rangés de 2 à $Refset$. Poser le sous ensemble $s = \{s_1, s_2, ...s_k\}$ pour $2 \leq k \leq |Refset|$.
   (Nous considérons que la méthode de génération de sous ensembles saute les sous ensembles pour lesquels les éléments considérés n'ont pas changé dans les itérations précédentes).

6. Appliquer la méthode de combinaison des solutions à s pour obtenir une ou plusieurs nouvelles solutions x_s.

7. Appliquer la méthode d'amélioration à x_s pour obtenir une solution améliorée x_s^* ;
   Si(x^* n'est pas dans $Refset$ et que la valeur de la fonction objectif de x_s^* est meilleure que la valeur de la fonction objectif du plus mauvais élément de $Refset_1$) **alors**

8. Ajouter x_s^* à $Refset_1$ et supprimer le mauvais élément en cours dans $Refset_1$ (le mauvais élément est celui qui a la mauvaise valeur de la fonction objectif).

9. **Faire** Nouveaux Eléments = VRAI.

Autre
Si (x_s^* n'est pas dans $Refset_2$ et $d_{min}(x_s^*)$ est supérieur à $d_{min}(x)$ pour une solution x dans $Refset_2$)**alors**

10. Ajouter x_s^* à $Refset_2$ et supprimer le mauvais élément en cours dans $Refset_2$.
(le mauvais élément est la solution s avec la valeur la plus petite de $d_{min}(x)$.

11. Faire Nouveaux Eléments = VRAI.
Fsi
Fsi
Fpour
Ftantque
Si($iter < Maxlter$) **alors**

12. Construire un nouvel ensemble P en utilisant la méthode de génération de diversification. Initialiser le processus de génération avec les solutions en cours de $Refset_1$. Les premières b_1 solutions dans le nouvel ensemble P sont les meilleurs b_1 solutions en cours dans $Refset$.
Fsi
Fpour

3.4 Evolution de Scatter Search et applications

3.4.1 Evolution de Scatter Search

La méthode de Scatter search a été introduite pour la première fois en 1977 [25], comme une heuristique pour la programmation en nombres entiers, basée sur des stratégies de gestion présentées lors d'une conférence qui s'est tenue à Austin, au Texas en 1967. Étant donné la popularité de ce que l'on appelle des algorithmes génétiques (AG), également introduite dans les années soixante-dix et également basée sur le maintien et l'évolution d'un ensemble de solutions, plusieurs articles (Glover, 1994a, 1994b, 1995)[26],[27],[28] ont été consacrés à clarifier leurs origines différentes, de perspectives Communes et de leurs potentialités. En revanche, la recherche dispersée a été conçue comme une extension de l'heuristique dans le domaine des mathématiques de relaxation, qui a été conçu pour la solution des problèmes de programmation en nombres entiers : relaxation des contraintes de remplacement.

Comme indiqué dans Laguna et Martí (2003)[41] Scatter search n'a jamais été appliquée ou discutée de nouveau jusqu'en 1990, date à laquelle elle a été réintroduite à l'EPFL Séminaire sur les méthodes de recherche opérationnelle et de l'intelligence artificielle (Lausanne, Suisse). Un article basé sur cette présentation a été publié en 1994 (Glover, 1994b)[27] dans lequel la mise en œuvre de nouveaux détails sont fournis et la gamme d'application est étendu aux problèmes non linéaires, binaires et problèmes

de permutation. La procédure est couplée avec la recherche Tabou (TS), en utilisant les formes d'adaptation et de l'aspiration de mémoire, critères d'influencer le choix des points de référence dans un ensemble composé de plusieurs sous-ensembles. Le concept des combinaisons pondérées est présenté comme le principal mécanisme permettant de générer de nouveaux points d'essai sur les lignes qui joignent les points de référence. De plus, il introduit les concepts de la combinaison des solutions de haute qualité avec les diverses solutions pour s'occuper des problèmes d'optimisation discrète.

Glover, Kelly et Laguna (1995)[32] décrivent la recherche dispersée comme un lien entre la recherche Tabou et l'algorithme génétique. Ils explorent la nature des liens entre ces méthodes et montrent une variété de possibilités pour la création d'approches hybrides. Glover (1995)[28] a proposé des concepts et des stratégies (surtout dans le contexte des combinaisons structurées) ne sont pas encore exploités dans la tradition génétique, et qui présentent des propriétés particulières pour l'exploitation des problèmes d'optimisation combinatoire

En 1998, un article a été publié sur la procédure de Scatter search par Glover [30]. Ce document présente une description de l'algorithmique et de la méthode de Scatter search, depuis de nombreuses applications ont été développées après. Cette version intègre beaucoup de la mise en œuvre et les détails algorithmiques qui ont suscité l'intérêt des chercheurs et des praticiens.

Elle consiste à générer un ensemble de départ de vecteurs solutions pour garantir un niveau critique de la diversité et applique les processus heuristiques conçus pour le problème considéré comme une tentative pour améliorer ces solutions. Puis, un sous-ensemble des meilleurs vecteurs (en termes de qualité et de diversité) est désigné pour être des solutions de référence. De nouvelles solutions sont créées au moyen des combinaisons des solutions de l'ensemble de référence et des processus heuristiques sont de nouveau utilisés pour améliorer les nouvelles solutions. Enfin, une collection de la "meilleure" des solutions améliorées est ajoutée à l'ensemble de référence. Ces étapes sont répétées jusqu'à ce que l'ensemble de référence ne change pas après deux itérations successives

Laguna et Martí (2003)[41] fournissent une description détaillée de la méthode de Scatter search pour trois classes de problèmes avec la représentation différentes des solutions : permutation des vecteurs, variables continues et variables binaires. Les auteurs ont également abordé les problèmes sans contraintes et contraint de fournir des implémentations spécifiques de la méthodologie à ces classes de problèmes.

3.4.2 Applications

Dans cette section, on trouve toutes les implémentations de la méthode de Scatter search développées jusqu'à présent, les documents publiés (et chapitres de livres) et les rapports techniques qui sont déjà été acceptés pour publication.
Le tableau (3.1) les classe en fonction des problèmes spécifiques ou des domaines d'application. Ainsi que les documents théoriques ou méthodologiques.

Description	Références
Méthodologie	Glover (1998), Binato et al.(2001), Glover (1994a-1999), Glover, Laguna and Marti (2003a-b, 2004a-b) Greistorfer(2004), Greistorfer and Voss (2004) Laguna (2002), Laguna and Martí(2003) ,Martí et al.(2004), Reeves and Yamada (1999), Resende and Ribeiro (2002, 2003b)
Affectation	Alfandari et al.(2001), Alfandari et al.(2004), Cung et al.(1997), Martí et al.(2000), Oliveira et al.(2003), Yagiura et al.(2002), Yagiura et al. (2004)
Problèmes binaires	Amini et al. (1999)
Clustering / Selection	Cotta (2004), García-López et al.(2004), Scheuerer and Wendolsky (2004), Ribeiro et al.(2003)
Coloration	Hamiez and Hao (2002)
Commercial Soft.	Laguna and Martí (2002)
Continus	Fleurent et al.(1996), Herrera et al.(2004), Trafalis et al.(1996), Ugray et al.(2002)
Graph Drawing	Laguna and Martí (1999)
Problèmes de graphes	Alvarez et al.(2001),Bastos et al.(2001), Cavique et al.(2001),Dell'Amico et al.(2004, Piñana et al.(2004),Souza et al.(2003), Xu et al.(2000), Festa et al.(2002), Ribeiro and Rosseti(2002), Ribeiro et al.(2002), Zhang and Lai(2004)
Knapsack	Da Silva et al.(2004), Díaz et al.(2004)
Linear Ordering	Campos et al.(1999), Campos et al.(2001)
Prog en entiers mixtes	Glover et al.(2000)
Multi Objective	Beausoleil (2001), Beausoleil (2004)
Neural Networks	Kelly et al.(1996),Laguna and Marti(2000), El-Fallahi and Martí (2003), El-Fallahi et al.(2004)
p-Median	Díaz and Fernández (2004), García-López et al.(2003), Pérez et al.(2004)
Problèmes de Permutation	Campos et al.(2003), Martí, Laguna and Campos (2004)
Routing	Chu et al. (2004), Corberán et al. (2002) Greistorfer (1999), Greistorfer (2001), Rego (2000), Resende and Ribeiro (2003a) Chiang and Russell(2004)
ordonnancement	Aiex et al. (2003), Debels et al. (2004), Nowicki and Smutnicki (2004a-b), Reeves and Yamada (1998), Yamashita et al. (2004)
Software Testing	Sagarna and Lozano (2004)

TAB. 3.1 – **Publications sur la méthode de scatter search** (Source : Rafael Marti (2004))

Cette figure classe les contributions dans le tableau 1 en fonction de l'année de publication, et montre l'impact de la méthodologie de Scatter Search, en donnant la fréquence des publications de 1994 jusqu'à 2004. Egalement la fréquence cumulée, ce qui montre l'augmentation spectaculaire du nombre de publications traitant du Scatter Search.

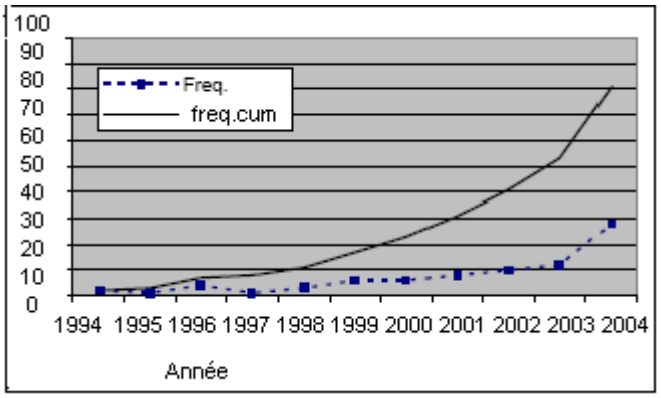

FIG. 3.1 – **Fréquence des publications de Scatter search**

Il convient aussi de mesurer l'impact global de la méthodologie de la Scatter Search, en considérant non seulement les documents publiés, mais aussi les rapports techniques, des présentations de conférence, et même des projets en cours de développement

Une première approche simple pour mesurer l'impact global d'une méthode pourrait consister à effectuer une recherche sur Internet. Quatre moteurs de recherche bien connus ont été utilisés : tout le web, Google, MSN et Yahoo et trois demandes : " optimisation par la recherche tabou ", " les algorithmes génétiques" et optimisation par "Scatter search". Le tableau (3.2) résume le nombre d'entrées trouvées par chaque moteur pour chaque requête. Nous pouvons voir que l'impact de la méthode de Scatter search est resté modéré en comparaison avec les deux autres procédures.

Moteur de recherche	Recherche Tabou	Algorithme génétique	scatter search
Tout le Web	7.180	48.962	737
Google	17.800	121.000	1.870
Msn	3.241	21.277	325
Yahoo	18.800	122.000	1.290

TAB. 3.2 – **Mesure de l'impact de scatter search**

L'objectif d'amélioration de la performance des métaheuristiques est souvent en conflit avec l'objectif de concevoir une méthode qui est facile à appliquer.
Les applications hybrides montrent que d'autres métaheuristiques peuvent être améliorées

lorsqu'elles sont combinées avec Scatter search.

Comme indiqué dans Laguna et Martí (2003)[41], l'élaboration et l'application des procédures métaheuristiques entraînent généralement une bonne dose d'expérimentation et en s'appuyant sur les expériences passées. Une méthode métaheuristique est basée sur des principes et non pas nécessairement sur la théorie qui peuvent être énoncés avec des théorèmes et des preuves. Par conséquent, l'objectif principal est de fournir des implémentations concrètes pour résoudre les problèmes difficiles d'optimisation.

Chapitre 4

Scatter Search pour le problème de tournées de véhicules

4.1 Introduction

Dans ce chapitre, nous allons illustrer la méthode de Scatter Search pour le problème de tournées de véhicules à travers un exemple, et nous avons modélisé ce problème sous forme d'un problème de permutation.[51]

Définition 4.1.1. Soit $G = (V, A)$ un graphe où $V = \{v_0, v_1, ...v_n\}$ est l'ensemble des sommets et $A = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$ est un ensemble d'arcs. Le sommet v_0 représente le dépôt, une flotte de m véhicules identiques de capacité Q où l'ensemble de sommets $V^{'} = V - \{v_0\}$ représente n villes (ou localisation des clients), une matrice des coûts ou de distances $C = (c_{ij})$ qui satisfait l'inégalité triangulaire $c_{ij} \leq c_{ik} + c_{kj}$ est définie sur A. Quand $c_{ij} = c_{ji}$ pour tout $(v_i, v_j) \in A$, le problème devient symétrique, alors on peut remplacer l'ensemble A par un ensemble d'arêtes $E = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$,on suppose que : $m \in [\underline{m}, \overline{m}]$ avec $\underline{m} = 1$ et $\overline{m} = n - 1$, les véhicules font des collections ou des livraisons mais pas les deux à la fois . Chaque sommet v_i lui est associé une quantité q_i,($q_0 = 0$) d'une certaine marchandise qui doit être livrée par un véhicule. Le (VRP) consiste à déterminer un ensemble m d'itinéraires de capacité minimale commençant par v_0 et se terminant en v_0. Chaque sommet $v_i \in V^{'}$ est visité une seule fois par un seul véhicule, la quantité totale attribuée à chaque itinéraire ne dépasse pas la capacité Q du véhicule qui assure le service de la route.

On définit une solution du VRP comme étant un ensemble de m routes, $S = \{R_1, R_2, ...R_m\}$, $R_k = (v_0, v_{R_1}, ..v_0)$, où R_k représente un ensemble ordonné de sommets consécutifs représentants l'itinéraire k

Le problème peut être formuler sous forme d'un problème de permutation comme suit :

$$min \sum_{k=1}^{m} \sum_{i=0}^{n_k} C(\pi_{i^k}, \pi_{i+1}^k) \tag{4.1}$$

$$\sum_{i=1}^{n_k} q_{\pi_i^k} \leq Q_k, k = 1, 2...m \tag{4.2}$$

$$\sum_{k=1}^{m} n_k = n \tag{4.3}$$

Une solution S est définie par la permutation $\left\{\pi_1^1, ..., \pi_{n_1}^m, ..\pi_{n_m}^m\right\}$ de sommets $\{v_1, v_2, ..v_n\}$,le sommet v_0 est implicitement inséré au début et à la fin de chaque tournée. π_i^k représente le sommet i affecté au véhicule k.

Considérons cette instance du problème de tournées de véhicules avec $n = 14$, $Q = 30$, $q = q_i$,$(i = 0, \ldots .14) = (0, 6, 20, 9, 8, 10, 8, 7, 5, 5, 4, 3, 4, 7)$, $m \in [1, 14]$, une matrice des coûts $C = (c_{ij})$,$(i, j = 1, \ldots .14, i \neq j)$ est définie comme suit :

$$\begin{pmatrix}
0 & 7.85 & 10.82 & 8.18 & 9.71 & 4.53 & 3.13 & 5.11 & 6.40 & 7.54 & 6.40 & 9.30 & 8.51 & 4.61 & 7.96 \\
 & 0 & 4.40 & 4.36 & 4.19 & 9.41 & 9.39 & 2.77 & 1.57 & 1.12 & 2.82 & 1.73 & 8.98 & 12.04 & 15.74 \\
 & & 0 & 10.89 & 12.52 & 15.24 & 13.62 & 12.48 & 13.04 & 14.96 & 15.02 & 16.10 & 19.19 & 13.20 & 13.54 \\
 & & & 0 & 1.84 & 11.47 & 10.80 & 4.40 & 3.78 & 5.34 & 6.48 & 5.76 & 12.59 & 12.79 & 15.88 \\
 & & & & 0 & 12.60 & 12.12 & 5.30 & 4.30 & 5.30 & 6.82 & 5.12 & 13.05 & 14.30 & 17.54 \\
 & & & & & 0 & 1.71 & 7.31 & 8.46 & 8.60 & 6.85 & 10.26 & 5.02 & 4.15 & 8.01 \\
 & & & & & & 0 & 6.96 & 8.22 & 8.75 & 7.14 & 10.48 & 6.60 & 2.94 & 6.91 \\
 & & & & & & & 0 & 1.30 & 2.72 & 2.60 & 4.34 & 8.27 & 9.44 & 13.04 \\
 & & & & & & & & 0 & 1.92 & 2.70 & 3.26 & 8.85 & 10.74 & 14.35 \\
 & & & & & & & & & 0 & 1.81 & 1.77 & 7.88 & 11.50 & 15.30 \\
 & & & & & & & & & & 0 & 3.40 & 6.24 & 9.98 & 13.86 \\
 & & & & & & & & & & & 0 & 8.95 & 13.26 & 17.07 \\
 & & & & & & & & & & & & 0 & 9.14 & 12.87 \\
 & & & & & & & & & & & & & 0 & 3.97 \\
 & & & & & & & & & & & & & & 0
\end{pmatrix}$$

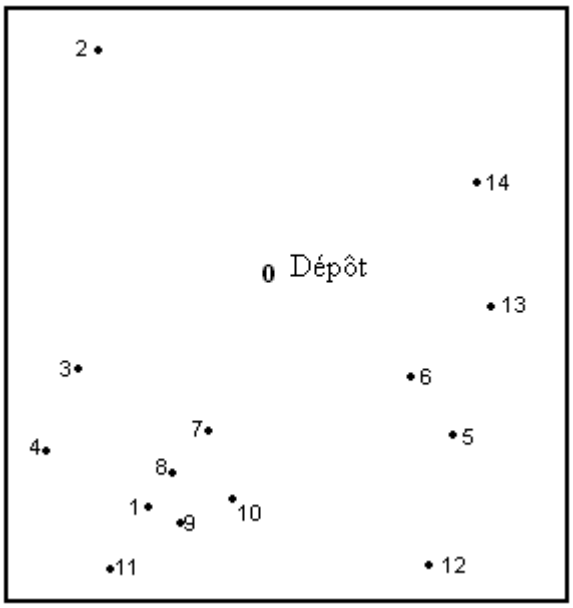

FIG. 4.1 – **Distribution des clients**

Maintenant, appliquons la méthode de Scatter Search à ce problème.

4.2 Méthode de génération de diversification

Scatter search commence par un ensemble de solutions généré de la manière suivante :

On choisit une permutation initiale donnée par :$P = \{1, 2, ...14\}$, soit h un entier tel que :$h \leq n = 14$

$P(h : s) = (s, s + h, s + 2h, s + 3h, ...s + rh)$ avec $s + rh \leq n$, $h \leq n$, et on définit la permutation $p(h)$ comme suit :$P(h) = P(h : h), P(h : h - 1), ..., P(h : 1)$.

Illustration

Pour $h = 4, P(4 : 4) = (4, 8, 12)$,$P(4 : 3) = (3, 7, 11)$, $P(4 : 2) = (2, 6, 10, 14)$,
$P(4 : 1) = (1, 5, 9, 13)$ d'où :
$P(4) = (4, 8, 12, 3, 7, 11, 2, 6, 10, 14, 1, 5, 9, 13)$.

Pour trouver les tournées, on procède de la manière suivante :

On construit R_k tel que $\sum q_i \leq 30 = Q$

Le résultat obtenu peut être considéré comme un processus généralisé d'affectation qui ne s'appuie pas sur l'ordre dans lequel les clients sont visités mais il veille à ce

$P(4)$	i	4	8	12	3	7	11	2	6	10	14	1	5	9	13
R_K	q_i	8	7	3	9	8	4	20	10	5	7	6	9	5	4
	R_1	8	15	18	27										
	R_2						8	12							
	R_3							20	30						
	R_4									5	12	18	27		
	R_5													5	9

TAB. 4.1 – **Détermination des routes**

que toutes les solutions qui peuvent être créées sont réalisables et différentes (car elles proviennent de différentes permutations), le parcours d'un véhicule peut être déterminé à l'aide de l'algorithme de 2-opt utilisé pour le problème de voyageur de commerce. Le principe de la méthode est d'examiner chaque paire d'arêtes non adjacentes (x_1, y_1), (x_2, y_2) du circuit, les remplacer par les deux nouvelles (x_1, y_2) et (x_2, y_1) de manière à avoir le coût du nouveau circuit inférieur.

Coût total du circuit obtenu sera réduit de Δ qui est égal à la somme des coûts des

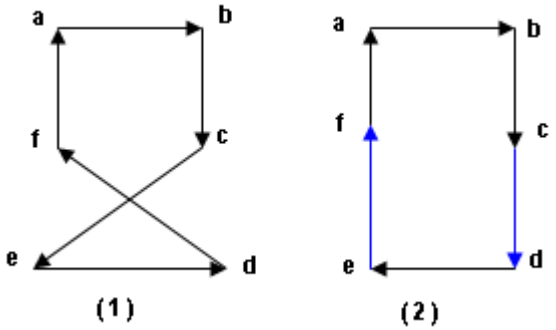

FIG. 4.2 – **application de la 2-opt**

arêtes échangées dans (1)-la somme des coûts des arêtes ajoutées dans (2). En appliquant cette procédure à l'exemple précédent, on obtient le résultat suivant :

Solution	Mouvement	R_1	R_2	R_3	R_4	R_5	Coût	Δ
$T_4^{(1)}$	v_0, v_{12}	4 8 12 3	7 11	2 6	10 14 1 5	9 13	163.54	-11.94
$T_4^{(2)}$	v_{10}, v_1	**12 8 4** 3	7 11	2 6	10 14 1 5	9 13	151.60	-12.45
$T_4^{(3)}$	v_1, v_5	12 8 4 3	7 11	2 6	10 **1 14** 5	9 13	139.15	-2.90
$T_4^{(4)}$	v_0, v_1	12 8 4 3	7 11	2 6	10 1 **5 14**	9 13	136.25	-1.10
S_4		12 8 4 3	7 11	2 6	**1 10** 5 14	9 13	135.15	

TAB. 4.2 – **Application de la 2-opt pour la solution T4**

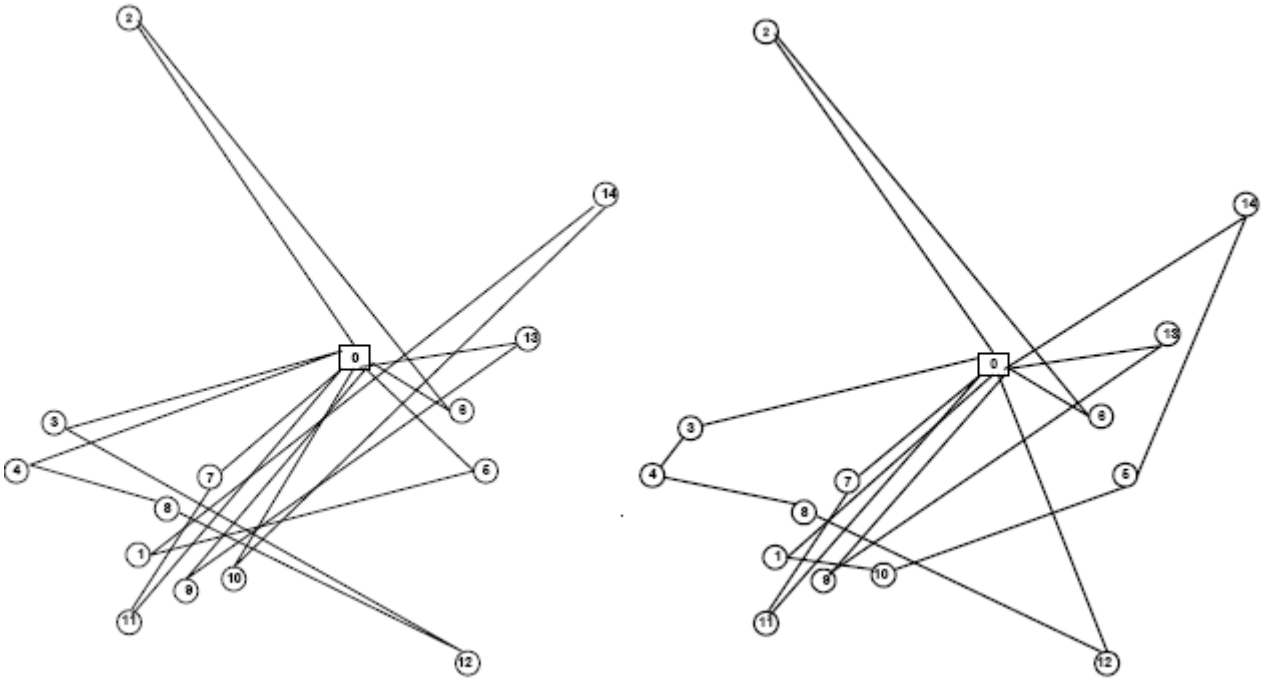

FIG. 4.3 – **Solutions T4 (gauche) et S4 (côté droit)**

4.3 Méthode d'amélioration

Considérons un processus itératif d'amélioration, une méthode basée sur la recherche locale. Cette méthode est conçue pour fonctionner directement sur le graphe du problème. La procédure consiste à enlever un sommet de sa position initiale et de l'insérer entre deux autres sommets

Solution	Routes	Coût
T_1	**0** 1 2 **0** 3 4 5 **0**6 7 8 9 **0** 10 11 12 13 14 **0**	120.90
S_1	**0** 1 2 **0** 3 4 5 **0** 6 9 8 7 **0** 11 10 12 14 13 **0**	120.83
T_2	**0** 2 4 **0** 6 8 10 12 **0** 14 1 3 **0**5 7 9 11 13 **0**	132.28
S_2	**0** 2 4 **0** 6 12 10 8 **0** 14 3 1 **0** 7 11 9 5 13 **0**	122.74
S_3	**0** 3 6 9 12 0 2 5 **0** 8 11 14 1 **0** 4 7 10 13 **0**	157 .24
T_3	**0** 6 12 9 3 **0** 2 5 **0** 8 1 11 14 **0** 7 4 10 13 **0**	128.27
T_4	**0** 4 8 12 3 **0** 7 11 **0** 2 6 **0** 10 14 1 5 **0** 9 13 **0**	163.54
S_4	**0** 12 8 4 3 **0** 7 11 **0** 2 6 **0** 1 10 5 14 **0** 9 13 **0**	135.15
T_5	**0** 5 10 4 9 **0** 14 3 8 13 **0** 2 7 **0** 12 1 6 11 **0**	149.08
S_5	**0** 5 10 9 4 **0** 8 3 14 13 **0** 2 7 **0** 1 11 12 6 **0**	119.50
T_6	**0** 6 12 5 11 **0** 4 10 3 9 **0** 2 8 **0** 14 1 7 13 **0**	140.97
S_6	**0** 6 5 12 11 **0** 3 4 9 10 **0** 2 8 **0** 14 13 1 7 **0**	113.74
T_7	**0** 7 14 6 13 **0** 5 12 4 11 **0** 3 10 **0** 2 9 **0** 1 8 **0**	139.83
S_7	**0** 7 6 13 14 **0** 5 12 11 4 **0** 3 10 **0** 2 9 **0** 1 8 **0**	130.47
T_8	**0** 8 7 6 **0** 14 5 13 4 **0** 3 11 **0** 2 10 **0** 1 9 **0**	146.83
S_8	**0** 7 8 6 **0** 14 13 5 4 **0** 12 11 3 **0** 2 10 **0** 1 9 **0**	136.29
T_9	**0** 9 8 7 6 **0** 5 14 4 13 **0** 3 12 **0** 2 11 1 **0** 10 **0**	148.42
S_9	**0** 7 8 9 6 **0** 14 13 5 4 **0** 3 12 **0** 2 11 1 **0** 10 **0**	137.16
T_{10}	**0** 10 9 8 7 **0** 6 5 4 **0** 14 3 13 **0** 2 12 1 **0** 11 **0**	148 .65
S_{10}	**0** 10 9 8 7 **0** 6 5 4 **0** 3 14 13 **0** 2 1 12 **0** 11 **0**	135.92

TAB. 4.3 – **Solutions finales produites par la méthode de génération de diversification**

Solution	Routes	Coût
S_1	**0** 2 **0** 3 4 5 **0** 6 9 8 7 **0** 1 11 10 12 14 13 **0**	109.67
S_2	**0** 2 **0** 3 4 1 8 **0** 6 5 13 14 **0** 7 11 9 10 12 **0**	92.51
S_3	**0** 6 5 12 9 **0** 2 **0** 8 1 11 13 14 **0** 7 3 4 10 **0**	106.37
S_4	**0** 8 4 3 **0** 7 1 11 9 10 **0** 2 **0** 12 5 6 **0** 13 14 **0**	96.84
S_5	**0** 5 12 10 9 4 **0** 7 3 14 13 **0**2 **0** 8 1 11 6 **0**	111.52
S_6	**0** 6 5 **0** 3 4 9 10 12 **0** 2 8 **0** 14 13 11 1 7 **0**	106.30
S_7	**0** 6 5 13 14 **0** 3 4 11 10 12 **0** 2 **0** 9 1 8 7 **0**	92.48
S_8	**0** 7 8 1 9 **0** 14 13 5 6 **0** 12 10 11 4 3 **0** 2 **0**	92.48
S_9	**0** 7 8 **0** 14 13 5 6 **0** 3 4 9 10 12 **0** 2 11 1 **0**	102.11
S_{10}	**0** 10 9 11 8 7 **0** 6 5 **0** 4 3 14 13 **0** 2 1 12 **0**	107.74

TAB. 4.4 – **Solutions améliorées**

4.4 Méthode de mise à jour de l'ensemble de référence

Cette méthode est utilisée pour créer et maintenir un ensemble de référence $Refset = Refset_1 \bigcup Refset_2$ avec $Refset_1$ est l'ensemble des meilleures solutions au sens de la fonction objectif et $Refset_2$ est l'ensemble des solutions diverses. La mesure de diversité est définie comme suit : $d_{ij} = |(S_i \bigcup S_j) \setminus (S_i \bigcap S_j)|$ comme une distance entre la solution S_i et S_j qui donne le nombre d'arêtes qui ne sont pas à la fois dans les deux solutions. La procédure consiste à calculer cette distance de chaque élément de $Refset_1$ vers toutes les autres solutions, choisir pour chacune la distance minimale et on sélectionne celle qui maximise ces distances comme élément de $Refset_2$.

Illustration
Dans cet exemple, on prend le cardinal de $|Refset| = b = 6$, avec $|Refset_1| = b_1 = 3$ et $|Refset_2| = b_2 = 3$
D'après le tableau précèdent, il est clair que les éléments de $Refet_1$ sont $\{S_7, S_2, S_4\}$ Le tableau suivant donne les résultats obtenus

Refset	Autres solutions					
	S_1	S_3	S_5	S_6	S_9	S_{10}
S_7	16	16	22	16	12	20
S_2	20	16	18	12	10	18
S_4	19	13	17	11	13	15
	16	13	**17**	11	10	15
S_5	18	16		16	18	18
	16	13		11	10	15
S_1		22		18	16	18
		13		11	10	**15**

TAB. 4.5 – **Les distances entre les solutions**

D'où $Refset_2 = \{S_5, S_1, S_{10}\}$, ainsi l'ensemble de référence
$Refset$=$\{S_7, S_2, S_4, S_5, S_1, S_{10}\}$.
On réordonne l'indice de ces solutions $Refset = \{S_1, S_2, S_3, S_4, S_5, S_6\}$,
nous obtenons le tableau suivant

4.5 Méthode de génération de sous ensembles

Dans cette étape, on construit les quatre types des sous ensembles de solutions de l'ensemble de référence de la manière dont ils sont définis dans la méthode de Scatter

Solution	Routes	Coût
S_1	**0** 6 5 13 14 **0** 3 4 11 10 12 **0** 2 **0** 9 1 8 7 **0**	92.48
S_2	**0** 2 **0** 3 4 1 8 **0** 6 5 13 14 **0** 7 11 9 10 12 **0**	92.51
S_3	**0** 8 4 3 **0** 7 1 11 9 10 **0** 2 **0** 12 5 6 **0** 13 14 **0**	96.84
S_4	**0** 5 12 10 9 4 **0** 7 3 14 13 **0** 2 **0** 8 1 11 6 **0**	111.52
S_5	**0** 2 **0** 3 4 5 **0** 6 9 8 7 **0** 1 11 10 12 14 13 **0**	109.67
S_6	**0** 10 9 11 8 7 **0** 6 5 **0** 4 3 14 13 **0** 2 1 12 **0**	107.74

TAB. 4.6 – **L'ensemble de référence initial**

Search pour les combiner à l'étape suivante.
Le tableau suivant nous donne ces sous ensembles.

Type	Sous ensembles
1	(S_1, S_2),(S_1, S_3),(S_1, S_4),(S_1, S_5) ,(S_1, S_6),(S_2, S_3), (S_2, S_4),(S_2, S_5),(S_2, S_6),(S_3, S_4),(S_3, S_5),(S_3, S_6), (S_4, S_5),(S_4, S_6),(S_5, S_6).
2	(S_1, S_2, S_4),(S_1, S_2, S_5),(S_1, S_2, S_6), (S_1, S_3, S_4),(S_1, S_3, S_5),(S_1, S_3, S_6),(S_1, S_4, S_5), (S_1, S_4, S_6),(S_1, S_5, S_6).
3	(S_1, S_2, S_3, S_4),(S_1, S_2, S_3, S_5),(S_1, S_2, S_3, S_6), (S_1, S_2, S_4, S_5),(S_1, S_2, S_4, S_6),(S_1, S_2, S_5, S_6).
4	$(S_1, S_2, S_3, S_5, S_6)$,$(S_1, S_2, S_3, S_4, S_5, S_6)$

TAB. 4.7 – **Génération des sous ensembles**

4.6 Méthode de combinaison

Cette méthode est désignée pour explorer les sous espaces de l'enveloppe convexe de l'ensemble de référence. On considère une solution de variables x_{ij} qui représente l'arête (v_i, v_j). De nouvelles solutions sont générées par les combinaisons linéaires pondérées qui sont structurées par les sous ensembles définis dans l'étape précédente.
Pour chaque sous ensemble E des types précédents, tel que $|E| = r$,on note par $H(E)$ l'enveloppe convexe de E. La solution $S \in H(E)$ est définie comme suit :

$$S = \sum_{t=1}^{r} \lambda_t S_t \tag{4.4}$$

$$\sum_{t=1}^{r} \lambda_t = 1 \tag{4.5}$$

$$\lambda_t \geq 0, t = 1, ...r \tag{4.6}$$

Où r est le nombre de solutions de chaque sous ensemble, où λ_t est donné par :

$$\lambda_t = \frac{\frac{1}{C(S_t)}}{\sum_{t=1}^{r} \frac{1}{C(S_t)}}$$

Puis on calcule le score de chaque variable x_{ij} comme suit :

$$score(x_{ij}) = \sum_{t=1}^{r} \lambda_t(x_{ij}^t)$$

x_{ij}^t signifie que x_{ij} est une arête dans la solution S_t, et les variables x_{ij} sont des valeurs binaires données de la manière suivante :

$$x_{ij} = \lfloor score(x_{ij}) + 0.5 \rfloor$$

Dans le tableau suivant une illustration de la combinaison du sous ensemble (S_1, S_2, S_3, S_4)

Solution	S_1	S_2	S_3	S_4		
λ_t	0,2643	0,2642	0,2524	0,2191		
Arêtes					$Score(x_{ij})$	x_{ij}
(1,4)		0,2642			0,2642	0
(1,7)			0,2524		0,2524	0
(1,8)	0,2643	0,2642		0,2191	0,7476	1
(1,9)	0,2643				0,2643	0
(1,11)			0,2524	0,2191	0,4715	0
(2,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(3,0)	0,2643	0,2642	0,2524		07809,	1
(3,4)	0,2643	0,2642	0,2524		0,7809	1
(3,7)				0,2191	0,2191	0
(3,14)				0,2191	0,2191	0
(4,0)				0,2191	0,2191	0
(4,8)			0,2524		0,2524	0
(4,9)				0,2191	0,2191	0
(4,11)	0,2643				0,2643	0
(5,0)				0,2191	0,2191	0
(5,6)	0,2643	0,2642	0,2524		0,7809	1
(5,12)			0,2524	0,2191	0,4715	0
(5,13)	0,2643	0,2642			0,5285	1
(6,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(6,11)				0,2191	0,2191	0
(7,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(7,8)	0,2643				0,2643	0
(7,11)		0,2642			0,2642	0
(8,0)		0,2642	0,2524	0,2191	0,7357	1
(9,0)	0,2643				0,2643	0
(9,10)		0,2642	0,2524	0,2191	0,7357	1
(9,11)		0,2642	0,2524		0,5166	1
(10,0)			0,2524		0,2524	0
(10,11)	0,2643				0,2643	0
(10,12)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(12,0)	0,2643	0,2642			0,5285	1
(13,0)			0,2524	0,2191	0,4715	0
(13,14)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(14,0)	0,2643	0,2642			0,5285	1

TAB. 4.8 – **Combinaison linéaire des solutions**

Une nouvelle solution est créée en représentant les arêtes dont les variables $x_{ij} = 1$. La combinaison linéaire de l'exemple précèdent donne les routes suivantes : (1,8,0), (2,0), (0,3,4) , (0, 6, 5, 13, 14,0),(0,11, 9, 10 ,12),(7,0).

Illustration

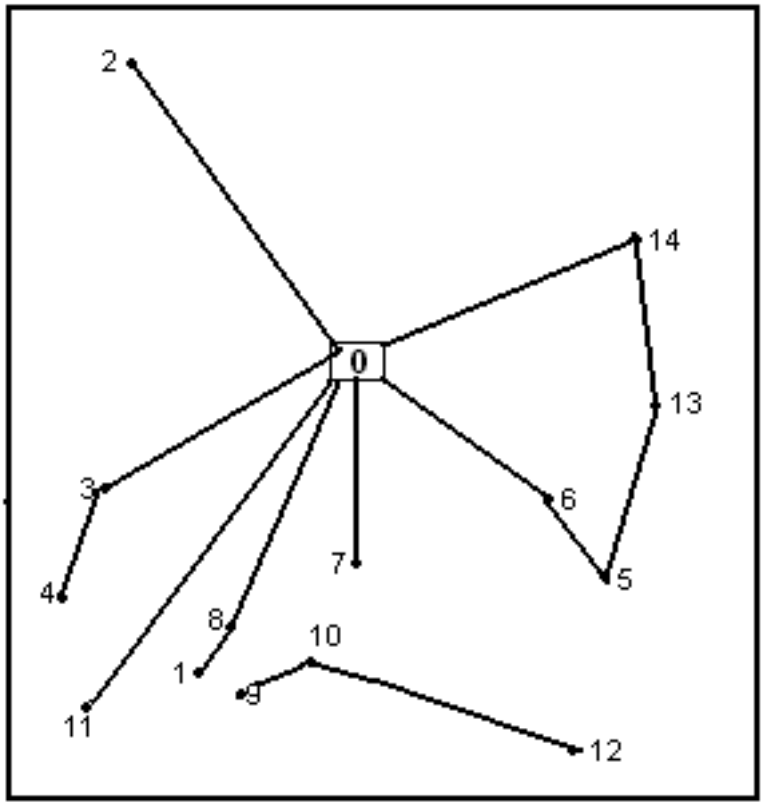

FIG. 4.4 – **Routes obtenues**

Après avoir représenté les arêtes, on relit les sommets qui sont de degré 1 au dépôt, ici, on a 1, 2, 4 et 7. Parfois on peut obtenir des sommets de degrés supérieur à 2, dans ce cas, on applique une technique qui consiste à annuler les arêtes ayant un score plus petit, jusqu'à avoir chaque sommet de degré 2.

4.7 Méthode d'amélioration

Une fois les solutions sont obtenues, on remarque que quelques unes sont répétées, on les supprime directement puis on applique à chacune une méthode d'amélioration, comme on l'a indiqué précédemment dans l'étape deux. Le tableau suivant nous donne un exemple d'application

Remarque 1. Cette solution n'est pas réalisable au début car on a $Q_3 = 35 > 30$

Solution Iteration	k	R_K	Q_k	$C(R_k)$	$C(C_{12})$
initial	1	0 7 11 9 10 0	28	19.58	
solution	2	0 2 0	20	21.60	
C_{12}	3	0 13 14 3 4 8 0	**35**	37.00	
	4	0 5 6 0	19	9.36	
	5	0 12 0	3	17.02	104.60
Iteration 1	Enlever	le sommet 14 de R_3 et	l'inserer dans R_2		
	1	0 7 1 11 9 10 0	28	19.58	
	2	0 **2 14 0**	27	32.32	
	3	0 **13 3** 4 8 0	28	29.94	
	4	0 5 6 0	19	9.36	
	5	0 12 0	3	17.02	108.23
Iteration 2	Enlever	le sommet 12 de R_5 et	l'inserer dans R_4		
	1	0 7 1 11 9 10 0	28	19.58	
	2	0 2 14 0	27	32.32	
	3	0 13 3 4 8 0	28	29.94	
	4	0 **12 5** 6 0	22	18.37	100.21
Itération 3	Enlever	le sommet 13 de R_3 et	l'inserer dans R_4		
Solution	1	0 7 1 11 9 10 0	28	19.58	
finale	2	0 2 14 0	27	32.32	
C^*_{12}	3	**0 3** 4 8 0	24	20.72	
	4	012 5 **6 13** 0	26	22.79	95.41

TAB. 4.9 – **Méthode d'amélioration**

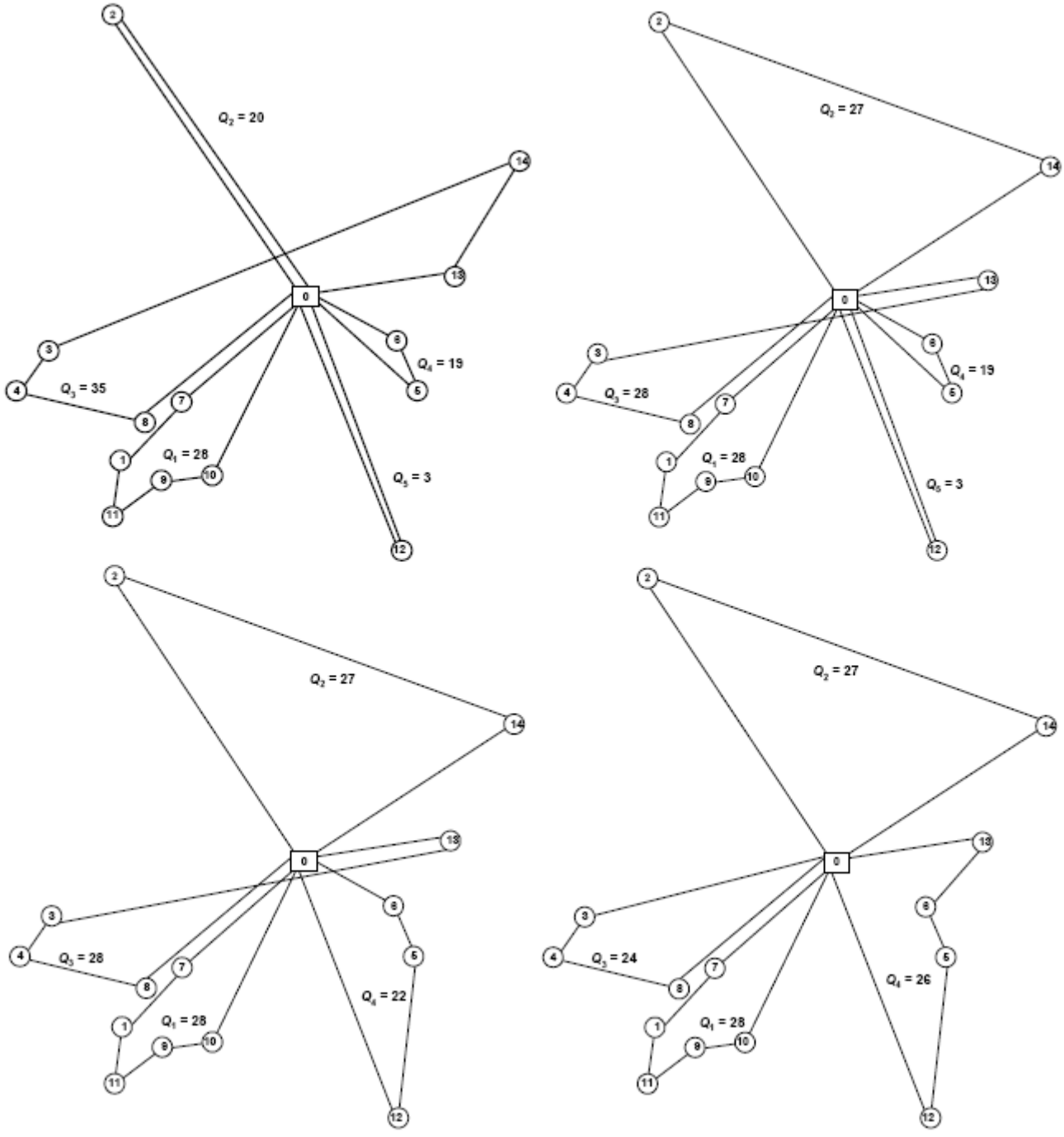

FIG. 4.5 – **Illustration graphique de la méthode d'amélioration**

Enfin, si une solution donne la meilleure valeur au sens de la fonction objectif par rapport aux éléments de $Refset_1$, on l'ajoute à cet ensemble et on supprime la mauvaise solution. De même si une solution donne la meilleure valeur au sens de diversité par rapport aux éléments de $Refset_2$, on l'ajoute à cet ensemble et on supprime la mauvaise. On répète la méthode de combinaison et d'amélioration jusqu'à ce que l'ensemble de référence ne change pas après deux itérations.
Dans cet exemple, on applique la méthode d'amélioration à toutes les autres solutions pour obtenir un ensemble d'éléments améliorés. La meilleure solution résulte de l'amélioration de C_{16} associé à une combinaison de sous-ensemble (S_1, S_2, S_3). C_{16} est la solution (0,1,8,0,2,0,3,4,0,6,5,13,14,0,7,0,12,10,9,11,0) avec un coût $C(c_{16}) = 115,94$, qui est transformé parla méthode d'amélioration dans la solution S avec un coût $C(c_{16}) = 91,01$. Cette instance du problème a été résolue en utilisant une méthode exacte (qui exige beaucoup plus de temps de calcul, même pour cet exemple simple) et il a été vérifié que S est en fait la solution optimale. Cette solution est illustrée dans la figure ci-dessous.

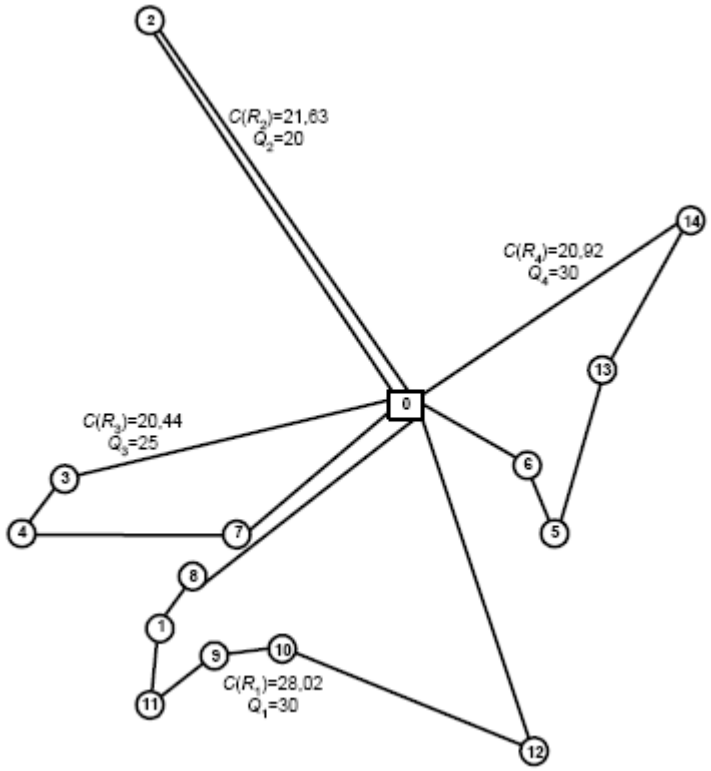

FIG. 4.6 – **Solution finale**

Chapitre 5

Scatter Search pour le problème de tournées de véhicules

5.1 Introduction

Dans ce chapitre, nous présentons la procédure de Scatter Search pour le problème de tournées de véhicules que nous avons modélisé sous forme d'un problème de permutation, et nous avons illustré la méthode en résolvant une petite instance.
La méthode de Scatter search a été introduite pour la première fois par Rochat en 1995[55]pour résoudre le problème de tournées de véhicules puis César Rego et Pedro Leao en 2001[51].

Définition 5.1.1. Soit $G = (V, A)$ un graphe où $V = \{v_0, v_1, ...v_n\}$ est l'ensemble des sommets et $A = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$ est un ensemble d'arcs. Le sommet v_0 représente le dépôt, une flotte de m véhicules identiques de capacité Q où l'ensemble de sommets $V^{'} = V - \{v_0\}$ représente n villes (ou localisation des clients), une matrice des coûts ou de distances $C = (c_{ij})$ qui satisfait l'inégalité triangulaire $c_{ij} \leq c_{ik} + c_{kj}$ est définie sur A. Quand $c_{ij} = c_{ji}$ pour tout $(v_i, v_j) \in A$, le problème devient symétrique, alors on peut remplacer l'ensemble A par un ensemble d'arêtes $E = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$,on suppose que : $m \in [\underline{m}, \overline{m}]$ avec $\underline{m} = 1$ et $\overline{m} = n - 1$, les véhicules font des collections ou des livraisons mais pas les deux à la fois. Chaque sommet v_i lui est associé une quantité q_i,$(q_0 = 0)$ d'une certaine marchandise qui doit être livrée par un véhicule. Le (VRP) consiste à déterminer un ensemble m d'itinéraires de capacité minimale commençant par v_0 et se terminant en v_0. Chaque sommet $v_i \in V^{'}$ est visité une seule fois par un seul véhicule, la quantité totale attribuée à chaque itinéraire ne dépasse pas la capacité Q du véhicule qui assure le service de la route.

On définit une solution du VRP comme étant un ensemble de m routes, $S = \{R_1, R_2, ...R_m\}$, $R_k = (v_0, v_{R_1}, ..v_0)$, où R_k représente un ensemble ordonné de sommets consécutifs représentants l'itinéraire k

Le problème peut être formuler sous forme d'un problème de permutation comme suit :

$$min \sum_{k=1}^{m} \sum_{i=0}^{n_k} C(\pi_i^k, \pi_{i+1}^k) \tag{5.1}$$

$$\sum_{i=1}^{n_k} q_{\pi_i^k} \leq Q_k, k = 1, 2...m \tag{5.2}$$

$$\sum_{k=1}^{m} n_k = n \tag{5.3}$$

Une solution S est définie par la permutation $\left\{\pi_1^1, ..., \pi_{n_1}^m, ..\pi_{n_m}^m\right\}$ de sommets $\{v_1, v_2, ..v_n\}$, le sommet v_0 est implicitement inséré au début et à la fin de chaque tournée. π_i^k représente le sommet i affecté au véhicule k.

5.2 Procédure de Scatter search

5.2.1 Méthode de génération de diversification

Scatter search commence par un ensemble de solutions généré de la manière suivante :

On choisit une permutation initiale donnée par : $p = \{1, 2, ...n\}$,
soit h un entier tel que :
$p(h : s) = (s, s + h, s + 2h, s + 3h, ...s + rh)$ avec $s + rh \leq n$ et $h < n$.
On définit la permutation $p(h)$ comme suit :
$p(h) = p(h : h), p(h : h - 1), ..., p(h : 1)$.

Pour trouver les tournées, on procède de la manière suivante :

On construit R_k, la route parcourue par le véhicule k tel que :
$Q_k = \sum_{v_i \in R_k} q_i \leq Q$

Le résultat obtenu peut être considéré comme un processus généralisé d'affectation qui ne s'appuie pas sur l'ordre dans lequel les clients sont visités mais il veille à ce que toutes les solutions qui peuvent être créées sont réalisables et différentes (car elles proviennent de différentes permutations), le parcours d'un véhicule peut être déterminé à l'aide de l'algorithme de 2-opt utilisé pour le problème de voyageur de commerce. Le principe de la méthode est d'examiner chaque paire d'arêtes non adjacentes (x_1, y_1), (x_2, y_2) du circuit, les remplacer par les deux nouvelles (x_1, y_2) et (x_2, y_1) de manière à avoir le coût du nouveau circuit inférieur.

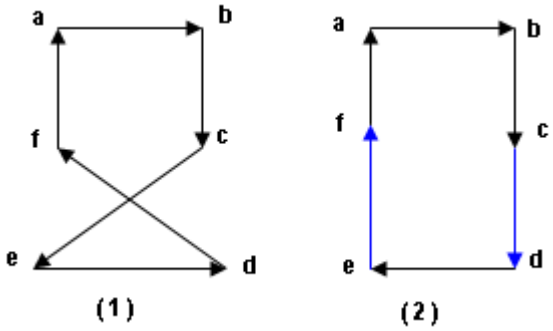

FIG. 5.1 – **application de la 2-opt**

Coût total du circuit obtenu sera réduit de Δ qui est égal à la somme des coûts des arêtes échangées dans (1)-la somme des coûts des arêtes ajoutées dans (2).

5.2.2 Méthode d'amélioration

Considérons un processus itératif d'amélioration, une méthode basée sur la recherche locale. Cette méthode est conçue pour fonctionner directement sur le graphe du problème. La procédure consiste à enlever un sommet de sa position initiale et de l'insérer entre deux autres sommets de manière à minimiser le coût des tournées, si on note par : $\underline{v}$ et $\overline{v}$ le prédécesseur et le successeur respectivement du sommet v, l'insertion du sommet v_i entre v_p et v_q revient à insérer les arêtes $(\underline{v_i}, \overline{v_i})$,$(v_p, v_i)$, (v_i, v_q) et en déplaçant $(\underline{v_i}, v_i)$, $(v_i, \overline{v_i})$ et $(v_p, \overline{v_p})$ où $v_q = \overline{v_p}$. Le coût de la solution devient :
$\Delta_{ip} = C(\underline{v_i}, \overline{v_i}) + C(v_p, v_i) + C(v_i, \overline{v_p}) - C(\underline{v_i}, v_i) - C(v_i, \overline{v_i}) - C(v_p, \overline{v_p})$.

5.2.3 Méthode de mise à jour de l'ensemble de référence

Cette méthode est utilisée pour créer et maintenir un ensemble de référence
$Refset = Refset_1 \bigcup Refset_2$ avec $|Refset| = b$, $|Refset_1| = b_1$ est l'ensemble de meilleures solutions au sens de la fonction objectif et $|Refset_2| = b_2$, est l'ensemble de solutions diverses. La mesure de diversité est définie comme suit : $d_{ij} = |(S_i \bigcup S_j) \setminus (S_i \bigcap S_j)|$ comme une distance entre la solution S_i et S_j qui donne le nombre d'arêtes qui ne sont pas à la fois dans les deux solutions. La procédure consiste à calculer cette distance de toutes le autres solutions vers chaque élément de $Refset_1$, choisir pour chacune la distance minimale et on sélectionne celle qui maximise ces distances comme élément de $Refset_2$.

5.2.4 Méthode de génération des sous ensembles

Dans cette étape, on construit les quatre types des sous ensembles de solutions de l'ensemble de référence de la manière suivante :

Type1 : Combiner les solutions de l'ensemble de référence $Refset$ deux à deux.

Type2 : Combiner les solutions de $Refset$ trois à trois en prenant chaque combinaison de type 1 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type3 : Combiner les solutions de $Refset$ quatre à quatre en prenant chaque combinaison de type 2 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type4 : Combiner les i meilleures solutions au sens de la fonction objectif de $Refset$ avec i variant de 5 à b

5.2.5 Méthode de combinaison

Cette méthode est désignée pour explorer les sous espaces de l'enveloppe convexe de l'ensemble de référence. On considère une solution de variables x_{ij} qui représente l'arête (v_i, v_j). De nouvelles solutions sont générées par les combinaisons linéaires pondérées qui sont structurées par les sous ensembles définis dans l'étape précédente.
Pour chaque sous ensemble E des types précédents, tel que $|E| = r$, on note par $H(E)$ l'enveloppe convexe de E. La solution $S \in H(E)$ est définie comme suit :

$$S = \sum_{t=1}^{r} \lambda_t S_t \tag{5.4}$$

$$\sum_{t=1}^{r} \lambda_t = 1 \tag{5.5}$$

$$\lambda_t \geq 0, t = 1, ...r \tag{5.6}$$

Où r est le nombre de solutions de chaque sous ensemble, et λ_t est donné par :

$$\lambda_t = \frac{\frac{1}{C(S_t)}}{\sum_{t=1}^{r} \frac{1}{C(S_t)}}$$

Puis on calcule le score de chaque variable x_{ij} comme suit :

$$score(x_{ij}) = \sum_{t=1}^{r} (\lambda_t x_{ij}^t)$$

x_{ij}^t signifie que x_{ij} est une arête dans la solution S_t, et les variables x_{ij} sont des valeurs binaires données de la manière suivante :

$$x_{ij} = \lfloor score(x_{ij}) + 0.5 \rfloor$$

Une nouvelle solution est créée en représentant les arêtes dont les variables $x_{ij} = 1$.
Après avoir représenté les arêtes, on relit les sommets qui sont de degré 1 au dépôt. Parfois on peut obtenir des sommets de degrés supérieur à 2, dans ce cas, on applique une technique qui consiste à annuler les arêtes ayant un score plus petit, jusqu'à avoir

chaque sommet de degré 2.

Une fois les solutions sont obtenues, on remarque que quelques unes sont répétées, on les supprime directement puis on applique à chacune une méthode d'amélioration, comme on l'a indiqué précédemment.

Enfin, si une solution donne la meilleure valeur au sens de la fonction objectif par rapport aux éléments de $Refset_1$, on l'ajoute à cet ensemble et on supprime la mauvaise solution. De même si une solution donne la meilleure valeur au sens de diversité par rapport aux éléments de $Refset_2$, on l'ajoute à cet ensemble et on supprime la mauvaise. On répète la méthode de combinaison et d'amélioration jusqu'à ce que l'ensemble de référence ne change pas après deux itérations successives.

5.3 Algorithme de Scatter Search pour le problème de tournées de véhicules

On définit les paramètres suivants :

$Psize$: la taille de l'ensemble de solutions générées par la méthode de génération de diversification.

b : cardinal de l'ensemble de référence.

b_1 : cardinal du sous ensemble de référence contenant les meilleures solutions au sens de la fonction objectif.

b_2 : cardinal du sous ensemble de référence contenant les solutions diverses.

$Maxlter$: nombre maximum d'itérations.

$Refset$: ensemble de référence.

$Refset_1$: sous ensemble contenant les meilleures solutions.

$Refset_2$: sous ensemble contenant les solutions diverses.

Algorithm 2 Scatter Search pour le problème de tournées de véhicules

1. Commencer par $P = \emptyset$. Soit n le nombre de clients. Utiliser une méthode de génération de diversification pour construire une solution S, en calculant :
   pour $h < n$, $p(h) = (p(h : h), p(h : h-1).....p(h : 1))$ avec : $p(h : s) = (s, s+h, s+2h....s+rh)$, $s + rh \leq n$
   Appliquer la méthode de la 2-opt à S. Utiliser la méthode d'amélioration pour obtenir une solution améliorée S^*.
   Si $S^* \notin P$, $P = P \bigcup S^*$, sinon omettre S^*. Répéter jusqu'à $|P| = PSize$.

2. Ordonner les solutions de P par rapport à la fonction objectif de la meilleure à la mauvaise solution.

   Pour(iter=1 **à** Maxlter)

3. Etablir $Refset = Refset_1 \bigcup Refset_2$ de P avec $|Refset| = b$, $|Refset_1| = b_1$, $|Refset_2| = b_2$.
   Prendre les premières solutions b_1 dans P et les ajouter à $Refset_1$. Pour chaque solution S_i dans $P - Refset$ et $S_j \in Refset$, calculer une mesure de distance de diversité

   $d_{ij} = |(S_i \bigcup S_j) \setminus (S_i \bigcap S_j)|$ qui donne le nombre d'arêtes qui ne sont pas à la fois dans les deux solutions.
   Choisir la solution $S^{'}$ qui maximise $d_{min}(S_i) = \underset{S_j \in refset}{min} \{d(S_i, S_j)\}$, ajouter $S^{'}$ à $Refset_2$, jusqu'à $Refset_2 = b_2$.
   Faire Nouveaux Eléments = VRAI.

   Tant que(Nouveaux Eléments) **Faire**

4. Calculer le nombre de sous ensembles (MaxSubset) qui incluent au moins un nouvel élément.
   Faire Nouveaux Eléments = FAUX.
   Pour(Subset =1,...., MaxSubset) **Faire**

5. Générer le sous ensemble $E(S)$ avec la méthode de génération de sous ensemble. Cette méthode génère l'un des quatre types des sous ensembles avec le nombre d'éléments rangés de 2 à $Refset$. Poser le sous ensemble $E(S) = \{S_1, S_2, ...S_k\}$ pour $2 \leq k \leq |Refset|$.
   (Nous considérons que la méthode de génération de sous ensembles saute les sous ensembles pour lesquels les éléments considérés n'ont pas changé dans les itérations précédentes).

6. Appliquer la méthode de combinaison des solutions à $E(S)$ pour obtenir une ou plusieurs nouvelles solutions $S^{'}$.

7. Appliquer la méthode d'amélioration à $S^{'}$ pour obtenir une solution améliorée $S^{'*}$;
Si($S^{'*}$ n'est pas dans $Refset$ et que la valeur de la fonction objectif de $S^{'*}$ est meilleure que la valeur de la fonction objectif du plus mauvais élément de $Refset_1$) **alors**

8. Ajouter $S^{'*}$ à $Refset_1$ et supprimer le mauvais élément en cours dans $Refset_1$ (le mauvais élément est celui qui a la mauvaise valeur de la fonction objectif).

9. **Faire** Nouveaux Eléments = VRAI.

Autre
Si($S^{'*}$ n'est pas dans $Refset_2$ et $d_{min}(S^{'*})$ est supérieur à $d_{min}(S)$ pour une solution S dans $Refset_2$)**alors**

10. Ajouter $S^{'*}$ à $Refset_2$ et supprimer le mauvais élément en cours dans $Refset_2$.

11. Faire Nouveaux Eléments = VRAI.
Fsi
Fsi
Fpour
Ftantque
Si($iter < Maxlter$) **alors**

12. Construire un nouvel ensemble P en utilisant la méthode de génération de diversification. Initialiser le processus de génération avec les solutions en cours de $Refset_1$. Les premières b_1 solutions dans le nouvel ensemble P sont les meilleurs b_1 solutions en cours dans $Refset$.
Fsi
Fpour

5.4 Exemple d'illustration

Pour illustrer cette méthode, nous considérons cette instance du problème de tournées de véhicules avec
$n = 14$, $Q = 30$, $q = q_i$, $(i = 0, \ldots .14) = (0, 6, 20, 9, 8, 10, 8, 7, 5, 5, 4, 3, 4, 7)$, $m \in [1, 14]$, une matrice des coûts $C = (c_{ij})$, $(i, j = 1, \ldots .14, i \neq j)$ est définie comme suit :

$$\begin{pmatrix}
0 & 7.85 & 10.82 & 8.18 & 9.71 & 4.53 & 3.13 & 5.11 & 6.40 & 7.54 & 6.40 & 9.30 & 8.51 & 4.61 & 7.96 \\
 & 0 & 4.40 & 4.36 & 4.19 & 9.41 & 9.39 & 2.77 & 1.57 & 1.12 & 2.82 & 1.73 & 8.98 & 12.04 & 15.74 \\
 & & 0 & 10.89 & 12.52 & 15.24 & 13.62 & 12.48 & 13.04 & 14.96 & 15.02 & 16.10 & 19.19 & 13.20 & 13.54 \\
 & & & 0 & 1.84 & 11.47 & 10.80 & 4.40 & 3.78 & 5.34 & 6.48 & 5.76 & 12.59 & 12.79 & 15.88 \\
 & & & & 0 & 12.60 & 12.12 & 5.30 & 4.30 & 5.30 & 6.82 & 5.12 & 13.05 & 14.30 & 17.54 \\
 & & & & & 0 & 1.71 & 7.31 & 8.46 & 8.60 & 6.85 & 10.26 & 5.02 & 4.15 & 8.01 \\
 & & & & & & 0 & 6.96 & 8.22 & 8.75 & 7.14 & 10.48 & 6.60 & 2.94 & 6.91 \\
 & & & & & & & 0 & 1.30 & 2.72 & 2.60 & 4.34 & 8.27 & 9.44 & 13.04 \\
 & & & & & & & & 0 & 1.92 & 2.70 & 3.26 & 8.85 & 10.74 & 14.35 \\
 & & & & & & & & & 0 & 1.81 & 1.77 & 7.88 & 11.50 & 15.30 \\
 & & & & & & & & & & 0 & 3.40 & 6.24 & 9.98 & 13.86 \\
 & & & & & & & & & & & 0 & 8.95 & 13.26 & 17.07 \\
 & & & & & & & & & & & & 0 & 9.14 & 12.87 \\
 & & & & & & & & & & & & & 0 & 3.97 \\
 & & & & & & & & & & & & & & 0
\end{pmatrix}$$

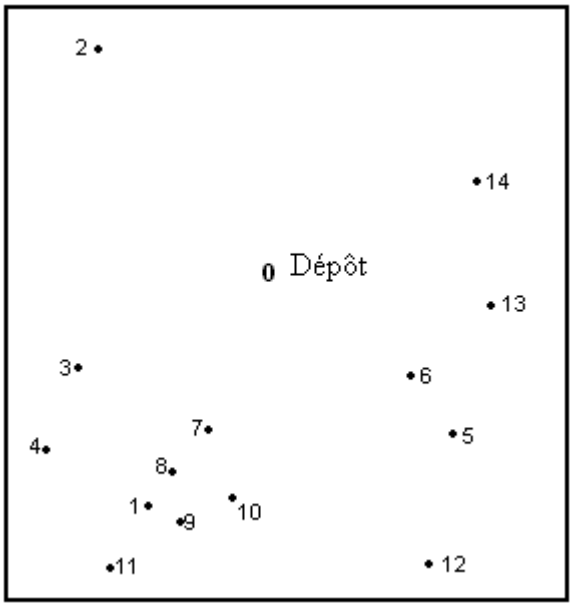

FIG. 5.2 – **Distribution des clients**

Maintenant, appliquons la méthode de Scatter Search à ce problème.
$P = \{1, 2, ...14\}$, soit h un entier tel que : $h < n = 14$
on calcule $P(h)$ pour $h = 1.....10$
Pour $h = 4, P(4:4) = (4, 8, 12)$,$P(4:3) = (3, 7, 11)$, $P(4:2) = (2, 6, 10, 14)$,
$P(4:1) = (1, 5, 9, 13)$ d'où :
$P(4) = (4, 8, 12, 3, 7, 11, 2, 6, 10, 14, 1, 5, 9, 13)$.

Pour trouver les tournées, on procède de la manière suivante :

$P(4)$	i	4	8	12	3	7	11	2	6	10	14	1	5	9	13
R_K	q_i	8	7	3	9	8	4	20	10	5	7	6	9	5	4
	R_1	8	15	18	27										
	R_2					8	12								
	R_3							20	30						
	R_4									5	12	18	27		
	R_5													5	9

TAB. 5.1 – **Détermination des routes**

On construit R_k tel que $Q_k = \sum_{v_i \in R_k} q_i \leq 30 = Q$

La méthode d'amélioration nous donne les résultats suivants :

Solution	Mouvement	R_1	R_2	R_3	R_4	R_5	Coût	Δ
$T_4^{(1)}$	v_0, v_{12}	4 8 12 3	7 11	2 6	10 14 1 5	9 13	163.54	-11.94
$T_4^{(2)}$	v_{10}, v_1	**12 8 4** 3	7 11	2 6	10 14 1 5	9 13	151.60	-12.45
$T_4^{(3)}$	v_1, v_5	12 8 4 3	7 11	2 6	10 **1 14** 5	9 13	139.15	-2.90
$T_4^{(4)}$	v_0, v_1	12 8 4 3	7 11	2 6	10 1 **5 14**	9 13	136.25	-1.10
S_4		12 8 4 3	7 11	2 6	**1 10** 5 14	9 13	135.15	

TAB. 5.2 – **Application de la 2-opt pour la solution T4**

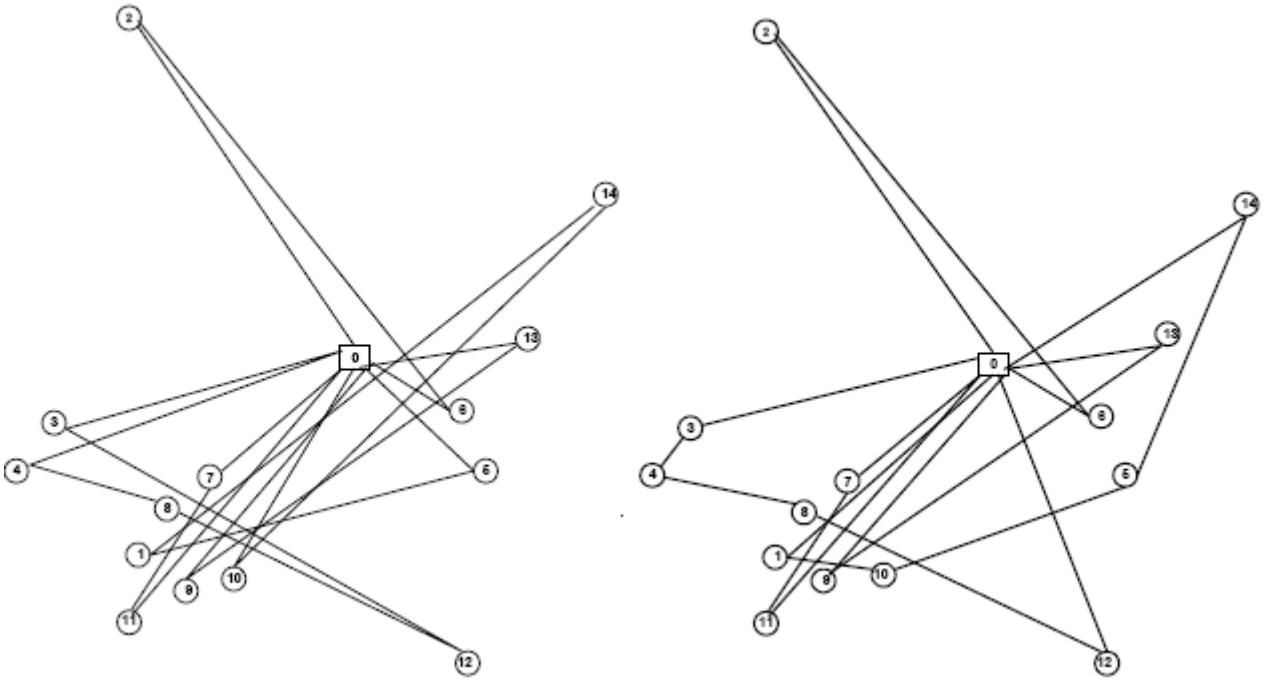

FIG. 5.3 – **Solutions T4(gauche)et S4 (côté droit)**

Solution	Routes	Coût
T_1	**0** 1 2 **0** 3 4 5 **0** 6 7 8 9 **0** 10 11 12 13 14 **0**	120.90
S_1	**0** 1 2 **0** 3 4 5 **0** 6 9 8 7 **0** 11 10 12 14 13 **0**	120.83
T_2	**0** 2 4 **0** 6 8 10 12 **0** 14 1 3 **0** 5 7 9 11 13 **0**	132.28
S_2	**0** 2 4 **0** 6 12 10 8 **0** 14 3 1 **0** 7 11 9 5 13 **0**	122.74
T_3	**0** 3 6 9 12 0 2 5 **0** 8 11 14 1 **0** 4 7 10 13 **0**	157 .24
S_3	**0** 6 12 9 3 **0** 2 5 **0** 8 1 11 14 **0** 7 4 10 13 **0**	128.27
T_4	**0** 4 8 12 3 **0** 7 11 **0** 2 6 **0** 10 14 1 5 **0** 9 13 **0**	163.54
S_4	**0** 12 8 4 3 **0** 7 11 **0** 2 6 **0** 1 10 5 14 **0** 9 13 **0**	135.15
T_5	**0** 5 10 4 9 **0** 14 3 8 13 **0** 2 7 **0** 12 1 6 11 **0**	149.08
S_5	**0** 5 10 9 4 **0** 8 3 14 13 **0** 2 7 **0** 1 11 12 6 **0**	119.50
T_6	**0** 6 12 5 11 **0** 4 10 3 9 **0** 2 8 **0** 14 1 7 13 **0**	140.97
S_6	**0** 6 5 12 11 **0** 3 4 9 10 **0** 2 8 **0** 14 13 1 7 **0**	113.74
T_7	**0** 7 14 6 13 **0** 5 12 4 11 **0** 3 10 **0** 2 9 **0** 1 8 **0**	139.83
S_7	**0** 7 6 13 14 **0** 5 12 11 4 **0** 3 10 **0** 2 9 **0** 1 8 **0**	130.47
T_8	**0** 8 7 6 **0** 14 5 13 4 **0** 3 11 **0** 2 10 **0** 1 9 **0**	146.83
S_8	**0** 7 8 6 **0** 14 13 5 4 **0** 12 11 3 **0** 2 10 **0** 1 9 **0**	136.29
T_9	**0** 9 8 7 6 **0** 5 14 4 13 **0** 3 12 **0** 2 11 1 **0** 10 **0**	148.42
S_9	**0** 7 8 9 6 **0** 14 13 5 4 **0** 3 12 **0** 2 11 1 **0** 10 **0**	137.16
T_{10}	**0** 10 9 8 7 **0** 6 5 4 **0** 14 3 13 **0** 2 12 1 **0** 11 **0**	148 .65
S_{10}	**0** 10 9 8 7 **0** 6 5 4 **0** 3 14 13 **0** 2 1 12 **0** 11 **0**	135.92

TAB. 5.3 – **Solutions finales produites par la méthode de génération de diversification**

Solution	Routes	Coût
S_1	**0** 2 **0** 3 4 5 **0** 6 9 8 7 **0** 1 11 10 12 14 13 **0**	109.67
S_2	**0** 2 **0** 3 4 1 8 **0** 6 5 13 14 **0** 7 11 9 10 12 **0**	92.51
S_3	**0** 6 5 12 9 **0** 2 **0** 8 1 11 13 14 **0** 7 3 4 10 **0**	106.37
S_4	**0** 8 4 3 **0** 7 1 11 9 10 **0** 2 **0** 12 5 6 **0** 13 14 **0**	96.84
S_5	**0** 5 12 10 9 4 **0** 7 3 14 13 **0**2 **0** 8 1 11 6 **0**	111.52
S_6	**0** 6 5 **0** 3 4 9 10 12 **0** 2 8 **0** 14 13 11 1 7 **0**	106.30
S_7	**0** 6 5 13 14 **0** 3 4 11 10 12 **0** 2 **0** 9 1 8 7 **0**	92.48
S_8	**0** 7 8 1 9 **0** 14 13 5 6 **0** 12 10 11 4 3 **0** 2 **0**	92.48
S_9	**0** 7 8 **0** 14 13 5 6 **0** 3 4 9 10 12 **0** 2 11 1 **0**	102.11
S_{10}	**0** 10 9 11 8 7 **0** 6 5 **0** 4 3 14 13 **0** 2 1 12 **0**	107.74

TAB. 5.4 – **Solutions améliorées**

Dans cet exemple, on prend le cardinal de $|Refset| = b = 6$, avec $|Refset_1| = b_1 = 3$ et $|Refset_2| = b_2 = 3$
D'après le tableau précèdent, il est clair que les éléments de $Refset_1$ sont $\{S_7, S_2, S_4\}$, calculons la distance entre les solutions de l'ensemble $\{S_1, S_3, S_5, S_6, S_9, S_{10}\}$ et celles de $Refset_1$, remarquons que la solution S_8 est omise de cet ensemble car elle est équivalente à S_7 qui appartient à $Refset_1$. Le tableau suivant donne les résultats obtenus

Refset	Autres solutions					
	S_1	S_3	S_5	S_6	S_9	S_{10}
S_7	16	16	22	16	12	20
S_2	20	16	18	12	10	18
S_4	19	13	17	11	13	15
	16	13	**17**	11	10	15
S_5	18	16		16	18	18
	16	13		11	10	15
S_1		22		18	16	18
		13		11	10	**15**

TAB. 5.5 – **Les distances entre les solutions**

D'où $Refset_2 = \{S_5, S_1, S_{10}\}$, ainsi l'ensemble de référence
$Refset$=$\{S_7, S_2, S_4, S_5, S_1, S_{10}\}$.
On réordonne l'indice de ces solutions $Refset = \{S_1, S_2, S_3, S_4, S_5, S_6\}$,
nous obtenons le tableau suivant

Solution	Routes	Coût
S_1	**0** 6 5 13 14 **0** 3 4 11 10 12 **0** 2 **0** 9 1 8 7 **0**	92.48
S_2	**0** 2 **0** 3 4 1 8 **0** 6 5 13 14 **0** 7 11 9 10 12 **0**	92.51
S_3	**0** 8 4 3 **0** 7 1 11 9 10 **0** 2 **0** 12 5 6 **0** 13 14 **0**	96.84
S_4	**0** 5 12 10 9 4 **0** 7 3 14 13 **0** 2 **0** 8 1 11 6 **0**	111.52
S_5	**0** 2 **0** 3 4 5 **0** 6 9 8 7 **0** 1 11 10 12 14 13 **0**	109.67
S_6	**0** 10 9 11 8 7 **0** 6 5 **0** 4 3 14 13 **0** 2 1 12 **0**	107.74

TAB. 5.6 – **L'ensemble de référence initial**

Maintenant, on construit les quatre types des sous ensembles de solutions de l'ensemble de référence pour les combiner à l'étape suivante.

Le tableau suivant nous donne ces sous ensembles.

Type	Sous ensembles
1	(S_1, S_2),(S_1, S_3),(S_1, S_4),(S_1, S_5) ,(S_1, S_6),(S_2, S_3), (S_2, S_4),(S_2, S_5),(S_2, S_6),(S_3, S_4),(S_3, S_5),(S_3, S_6), (S_4, S_5),(S_4, S_6),(S_5, S_6).
2	(S_1, S_2, S_3),(S_1, S_2, S_4),(S_1, S_2, S_5),(S_1, S_2, S_6), (S_1, S_3, S_4),(S_1, S_3, S_5),(S_1, S_3, S_6),(S_1, S_4, S_5), (S_1, S_4, S_6),(S_1, S_5, S_6).
3	(S_1, S_2, S_3, S_4),(S_1, S_2, S_3, S_5),(S_1, S_2, S_3, S_6), (S_1, S_2, S_4, S_5),(S_1, S_2, S_4, S_6),(S_1, S_2, S_5, S_6).
4	$(S_1, S_2, S_3, S_5, S_6)$,$(S_1, S_2, S_3, S_4, S_5, S_6)$

TAB. 5.7 – **Génération des sous ensembles**

On considère une solution de variables x_{ij} qui représente l'arête (v_i, v_j).

Dans le tableau suivant une illustration de la combinaison du sous ensemble (S_1, S_2, S_3, S_4)

Solution	S_1	S_2	S_3	S_4		
λ_t	0,2643	0,2642	0,2524	0,2191		
Arêtes					$Score(x_{ij})$	x_{ij}
(1,4)		0,2642			0,2642	0
(1,7)			0,2524		0,2524	0
(1,8)	0,2643	0,2642		0,2191	0,7476	1
(1,9)	0,2643				0,2643	0
(1,11)			0,2524	0,2191	0,4715	0
(2,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(3,0)	0,2643	0,2642	0,2524		0,7809	1
(3,4)	0,2643	0,2642	0,2524		0,7809	1
(3,7)				0,2191	0,2191	0
(3,14)				0,2191	0,2191	0
(4,0)				0,2191	0,2191	0
(4,8)			0,2524		0,2524	0
(4,9)				0,2191	0,2191	0
(4,11)	0,2643				0,2643	0
(5,0)				0,2191	0,2191	0
(5,6)	0,2643	0,2642	0,2524		0,7809	1
(5,12)			0,2524	0,2191	0,4715	0
(5,13)	0,2643	0,2642			0,5285	1
(6,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(6,11)				0,2191	0,2191	0
(7,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(7,8)	0,2643				0,2643	0
(7,11)		0,2642			0,2642	0
(8,0)		0,2642	0,2524	0,2191	0,7357	1
(9,0)	0,2643				0,2643	0
(9,10)		0,2642	0,2524	0,2191	0,7357	1
(9,11)		0,2642	0,2524		0,5166	1
(10,0)			0,2524		0,2524	0
(10,11)	0,2643				0,2643	0
(10,12)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(12,0)	0,2643	0,2642			0,5285	1
(13,0)			0,2524	0,2191	0,4715	0
(13,14)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(14,0)	0,2643	0,2642			0,5285	1

TAB. 5.8 – **Combinaison linéaire des solutions**

Une nouvelle solution est créée en représentant les arêtes dont les variables $x_{ij} = 1$. La combinaison linéaire de l'exemple précèdent donne les routes suivantes : (1,8,0), (2,0), (0,3,4) , (0, 6, 5, 13, 14,0),(0,11, 9, 10 ,12),(7,0).

Illustration

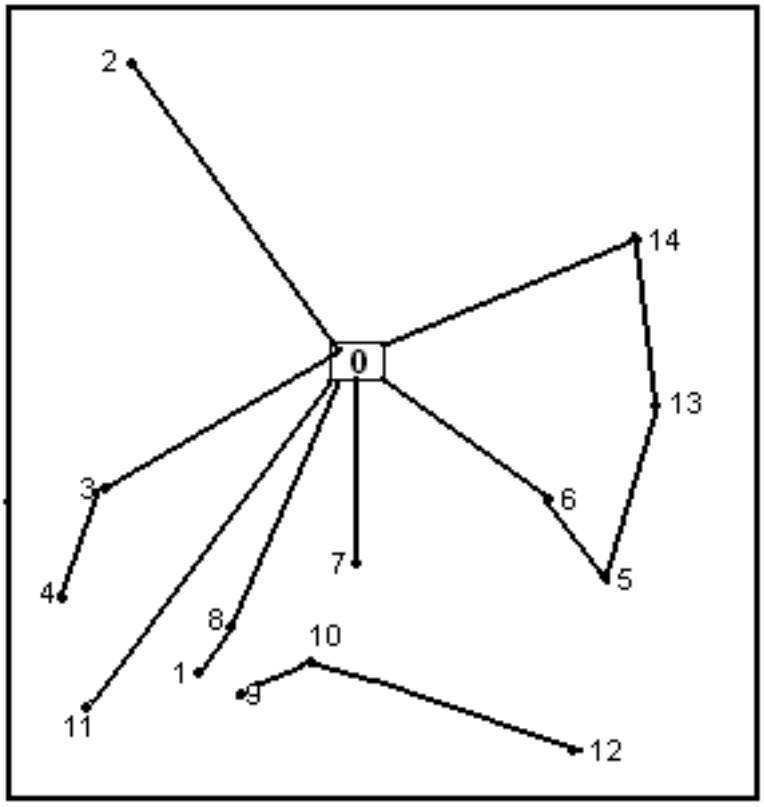

FIG. 5.4 – **Routes obtenues**

Après avoir représenté les arêtes, on relit les sommets qui sont de degré 1 au dépôt, ici, on a 1, 2, 4 et 7.

Une fois les solutions sont obtenues, on applique à chacune une méthode d'amélioration, comme on l'a indiqué précédemment dans l'étape deux. Dans cet exemple, on obtient 33 nouvelles solutions (une par chaque sous ensemble de combinaison). Notons par $C_1, C_2, ..., C_{33}$ ces solutions. Les solutions $C_{26}, .., C_{31}$ et C_{33} se répètent, on les supprime. Pour illustrer cette étape, nous avons choisi la solution C_{12} et on a montré comment appliqué la méthode d'amélioration, la solution C^*_{12} a été trouvé après trois itérations. Le tableau suivant nous donne les résultats.

Solution Iteration	k	R_K	Q_k	$C(R_k)$	$C(C_{12})$
initial	1	0 7 11 9 10 0	28	19.58	
solution	2	0 2 0	20	21.60	
C_{12}	3	0 13 14 3 4 8 0	**35**	37.00	
	4	0 5 6 0	19	9.36	
	5	0 12 0	3	17.02	104.60
Iteration 1	Enlever	le sommet 14 de R_3 et	l'inserer dans R_2		
	1	0 7 1 11 9 10 0	28	19.58	
	2	0 **2 14 0**	27	32.32	
	3	0 **13 3** 4 8 0	28	29.94	
	4	0 5 6 0	19	9.36	
	5	0 12 0	3	17.02	108.23
Iteration 2	Enlever	le sommet 12 de R_5 et	l'inserer dans R_4		
	1	0 7 1 11 9 10 0	28	19.58	
	2	0 2 14 0	27	32.32	
	3	0 13 3 4 8 0	28	29.94	
	4	0 **12 5** 6 0	22	18.37	100.21
Itération 3	Enlever	le sommet 13 de R_3 et	l'inserer dans R_4		
Solution	1	0 7 1 11 9 10 0	28	19.58	
finale	2	0 2 14 0	27	32.32	
C^*_{12}	3	**0 3** 4 8 0	24	20.72	
	4	012 5 **6 13** 0	26	22.79	95.41

TAB. 5.9 – **Méthode d'amélioration**

Remarque 2. Cette solution n'est pas réalisable au début car on a $Q_3 = 35 > 30$

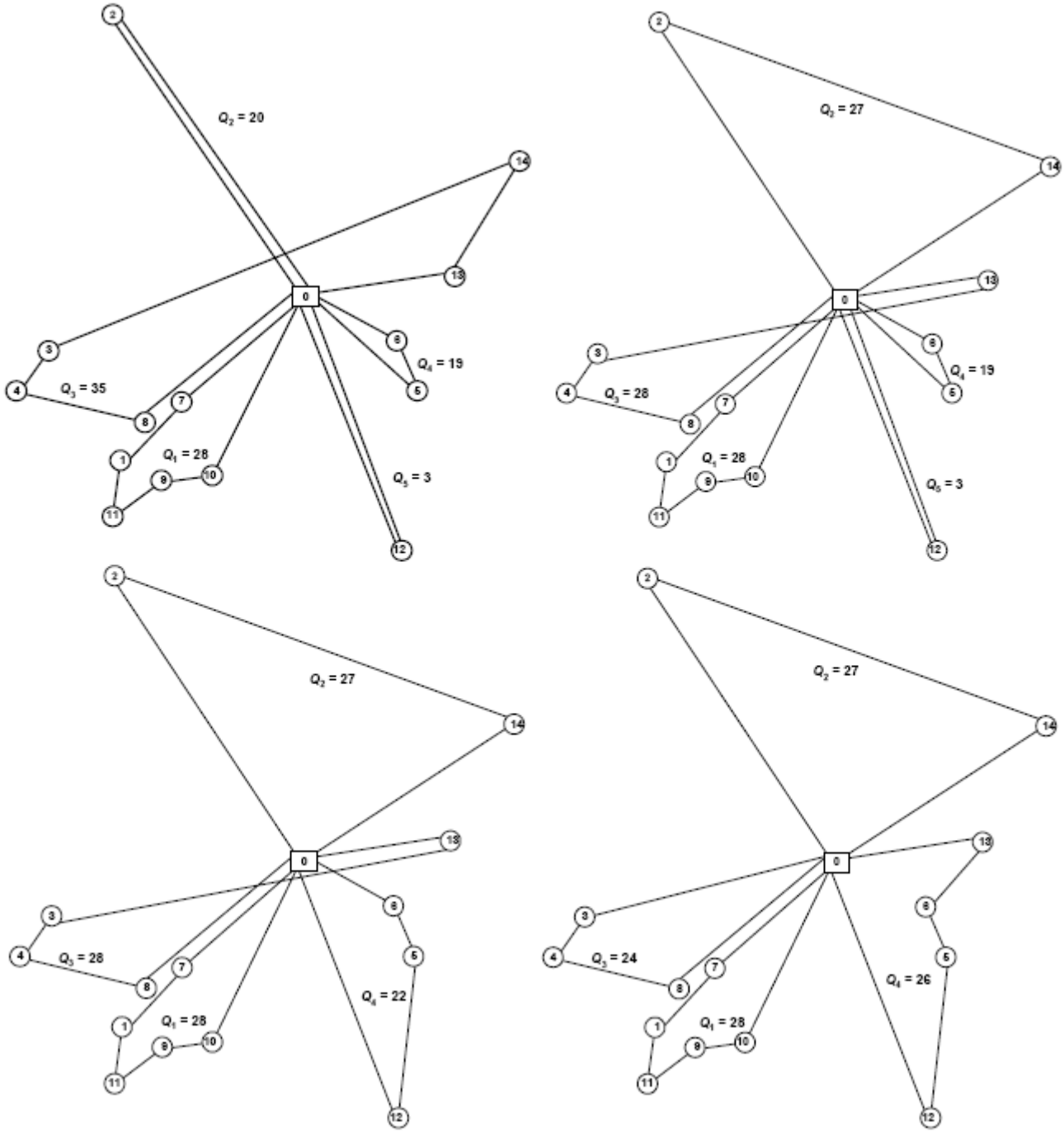

FIG. 5.5 – **Illustration graphique de la méthode d'amélioration**

On répète la méthode de combinaison et d'amélioration jusqu'à ce que l'ensemble de référence ne change pas après deux itérations.

Dans cet exemple, on applique la méthode d'amélioration à toutes les autres solutions pour obtenir un ensemble d'éléments améliorés. La meilleure solution résulte de l'amélioration de C_{16} associé à une combinaison de sous-ensemble (S_1, S_2, S_3). C_{16} est la solution (0,1,8,0,2,0,3,4,0,6,5,13,14,0,7,0,12,10,9,11,0) avec un coût $C(c_{16}) = 115,94$, qui est transformé par la méthode d'amélioration à la solution S_{16} avec un coût $C(S_{16}) = 91,01$. Cette instance du problème a été résolue en utilisant une méthode exacte (qui exige beaucoup plus de temps de calcul, même pour cet exemple simple) et il a été vérifié que S_{16} est en fait la solution optimale. Cette solution est illustrée dans la figure ci-dessous.

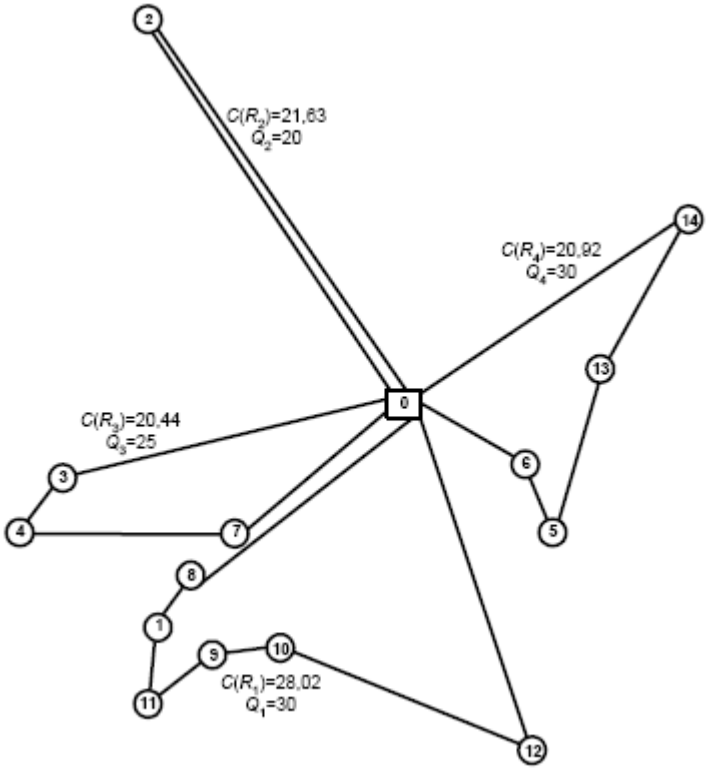

FIG. 5.6 – **Solution finale**

Conclusion et perspectives

Dans notre mémoire, nous avons décris la méthode de Scatter search. Cette approche évolutionnaire a pour origine les stratégies de création de règle de décision et de contraintes de substitution.

Cette méthode contraste avec d'autres procédures évolutionnaires, telles que les algorithmes génétiques, en utilisant des conceptions stratégiques là où d'autres approches utilisent l'aléatoire.

Des études récentes ont montré l'intérêt pratique de cette méthode pour résoudre divers problèmes d'optimisation, à cet effet, nous avons présenté une revue des travaux réalisés utilisant cette approche.

Nous nous sommes intéressés à l'application de Scatter search au problème de tournées de véhicules, vu l'importance des problèmes pratiques qui peuvent être modélisés sous forme d'un problème de VRP.

La recherche dispersée peut être utilisée pour résoudre d'autres variantes de problèmes de tournées de véhicules.

Cette approche offre diverses possibilités d'utilisation de mémoire adaptative comme base pour la création d'algorithmes efficaces pour la résolution de cas pratiques.

Bibliographie

[1] B. Adenso-Diaz, S. Garcia-Carvajal, and S. Lozano. An empirical investigation on parallelization strategies for scatter search. *European Journal of Operational Research*, 169(2) :490–507, 2006.

[2] M. Aiden and O. Oukacha. *Recherche opérationnelle Programmation Linéaire.*

[3] A. Alvarez, J.L. Gonzalez-Velarde, and K. De Alba. Grasp embedded scatter search for the multicommodity capacitated network design problem. *Journal of Heuristics*, 11(3) :233–257, 2005.

[4] A. Alvarez, J.L. Gonzalez-Velarde, and K. De Alba. Scatter search for network design problem,. *Annals of Operations Research*, 138(1) :159–178, 2005.

[5] E. Beltrami and L. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1) :65–94, 1974.

[6] B. Bennett and D. Gazis. School bus routing by computer. *Transportation Research*, 16(4) :317–325, 1972.

[7] J. Berger and M. Barkaoui. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Operations Research*, 2003.

[8] L. Bodin and B. Golden. Classification in vehicle routing and sheduling. *Networks*, 11 :97–108, 1981.

[9] L. Bodin, B. Golden, and M. Ball. Routing and sheduling of vehicles and crews : the state of the art. *Computers and operations research*, 10 :63–212, 1983.

[10] N. Christofides. The travelling salesman problem. *willey Chichester*, 1985.

[11] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operations Research quarterly*, 20 :309–318, 1969.

[12] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. *Combinatorial Optimiation, Wiley*, 11 :315–338, 1979.

[13] G. Clark and J. W. Wright. Sheduling of vehicles routing from a central depot to a number of delivery points. *Operations Research*, 12 :568–581, 1964.

[14] J. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. Technical report, Centre de recherche sur les transports, Montréal, Canada., 2000.

[15] J. M. Cordeau, M. Gendreau, G. Laporte, J. Y. Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, (53) :512–522, 2002.

[16] O. Cordon, S. Damas, J. Santamaria, and R. Marti. Scatter search for the 3d point matching problem in image registration. *INFORMS Journal on Computing, in press*, 2007.

[17] G. Dantzing and J. Ramser. The truck dipatching problem. *Operations research*, 12 :81–91, 1959.

[18] M. Desrochers, J. Desrosiers, and M. M. Solmon. A new optimisation algorithm for the vehicle routing problem with time windows. *Operations Research*, 140 :342–354, 1992.

[19] J. Egea, M. Rodriguez, J. Banga, and R. Marti. Scatter search for chemical and bio-process optimization. *Journal of Global Optimization*, 37(3) :481–503, 2007.

[20] M. L. Fisher and R. Jaikumar. A generalised assignement heuristic for vehicle routing. *Networks*, 11 :109–124, 1981.

[21] W. Garvin, H. Grandall, J. John, and R. Spellman. Application of linear programming in the oil industry. *Management Science*, 3(4) :407–430, Juillet 1957.

[22] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, (40) :1276–1290, 1994.

[23] F. Glover. A multiphase dual algorithm for the zero-one integer programming problem. *Operations Research*, 13(879), 1965.

[24] F. Glover. Surrogate constraint duality in mathematical programming. *Operations research*, (23) :434–451, 1975.

[25] F. Glover. Heuritics for integer programming using surrogate constraints. *Decision science*, 8 :156–166, 1977.

[26] F. Glover. Genetic algorithms and scatter search. *Statistics and computing*, 4 :131–140, 1994.

[27] F. Glover. Tabu search for nonlinear and parametric optimization scatter search. *Discrete Applied mathematic*, 49 :231–255, 1994.

[28] F. Glover. Scatter search and star paths : Beyong the genetic metaphor. *OR pektrum*, 17 :125–137, 1995.

[29] F. Glover. A template for scatter search and path relinking. *Lecture Notes in Computer Science*, (1363) :13–54, fevrier 1997.

[30] F. Glover. Genetic algorithms, evolutionary algorithms and scatter search : changing tides and untapped potentials. *INFORMS Computer Science Technical Section Newletter*, 1(19) :7–14, 1998.

[31] F. Glover. Scatter search and path relinking. *New Ideas in Optimiation*, pages 297–316, Fevrier 1999.

[32] F. Glover, J. Kelly, and M. Laguna. Genetic algorithms and tabu search : Hybrids for optimization. *Computers and Operations Research*, 1(22) :111–134, 1995.

[33] F. Glover, M. Laguma, and R. Marti. Scatter search. 2000.

[34] F. Glover, M. Laguna, and R. Marti. New ideas and applications of scatter search and path relinking. *New Optimization Techniques in Engineering, Springer-Verlag, forthcoming*, 2004.

[35] F. Glover, M. Laguna, and R. Marti. Scatter search and path relinking : Foundations and advanced designs. *New Optimization Techniques in Engineering, Springer-Verlag, forthcoming*, 2004.

[36] B. Golden. Vehicle routing problem formulation and heuristics studies techniques. Rapport Technique 113, 1975.

[37] B. Golden and A. Assad. Vehicle routing methods and studies. Rapport technique, North Holland, 1988.

[38] L. B. Golden and A. E. Wasil. Computerized vehicle routing in the soft drink industry. *Operations Research*, 1(35) :6–17, 1987.

[39] L. Jourdan. Métaheuristique pour l'extraction des connaissances. thèse de doctorat, 2003.

[40] M. Laguna. Scatter search. page 14, 1999.

[41] M. Laguna and R. Marti. Scatter search. methodology and implementations in c. *Kluwer Academic Publishers*, 2003.

[42] G. Laporte. Recent algorithmic developments for the traveling salesman problem and the vehicle routing problem. *Ricerca Operativa*, 23(68) :5–27, 1993.

[43] G. Laporte, M. Gendreau, J. Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, (7) :285–300, 2000.

[44] G. Laporte and Y. Nobert. Exact algorithms for vehicle routing problem. *Annals of discrete mathematics*, 31 :147–184, 1987.

[45] G. Laporte and I. H. Osman. Routing problems : A bibliography. *Annals of Operations Research*, (61) :227–262, 1995.

[46] J. K. Lenstra and Kan. A complexity of vehicle routing and scheduling problems. *Net Works*, 11 :221–227, 1981.

[47] H. Li and A. Lim. Local search with annealing-like restarts to solve vrptw. *European Journal of Operational Research*, 2003.

[48] J. A. Nelder and R. Mead. A simplex method for function minimisation. *Computer Journal*, 7(308), 1965.

[49] A. Parker, R. Dean, and R. Holmes. On the use of the vehicle routing algorithm for the parallel processor problem with sequence dependent changeover cost. *ALLE Transaction*, 9 :115–160, 1977.

[50] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. Research report, University of Technology of Troyes, 2001.

[51] C. Rego and P. Leao. A scatter search tutorial for graph-based permutation problems. November 2001.

[52] C. Rego and C. Roucairol. Le problème de tournées de véhicules, étude et résolution approchée. Rapport de recherche 2197, INRIA, 1994.

[53] C. Rego and C. Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. *Meta-Heuristics*, 1996.

[54] J. Renaud and F. Boctor. A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 3(140) :618–628, 2002.

[55] Y. Rochat and E. D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1 :147–167, 1995.

[56] D. J. Rosekrantz, R. E. Stearns, and P. M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3) :563–581, 1977.

[57] M. Sakarovitch. *Optimisation Combinatoire, Méthodes Mathématiques et Algorithmiques*. 1984.

[58] B. Sanso, F. Soumis, and M. Gendreau. Routing models and applications in telecommunications. *Net Works*, 1988.

[59] E. D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 1993.

[60] E. D. Taillard, P. Badeau, M. Gendreau, F. Geurtin, and J. Potvin. A tabu search heuristic for the vehicle routing problem with time windows. *Transportation Science*, (31) :170–186, 1997.

[61] P. Toth and D. Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 1-3(123) :487–512, 2002.

[62] A. Wahab, R. N. Monmarché, M. Slimane, M. Fahdil, and H. Saleh. A scatter search algorithm for the automatic clustering problem. *Lecture Notes in Artificial Intelligence*, pages 350–364, July 2006.

[63] www. wikipedia. *org*.

[64] J. Xu and J. Kelly. A network flow-based tabu search heuristics for the vehicle routing problem. *Transportation Science*, (30) :379–393, 1996.