République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahleb Blida
Département d'Informatique

Mémoire présenté pour l'obtention du diplôme de master

Professionnelle en système informatique et

Réseaux

« SIR »

MEMOIRE DE FIN D'ETUDE

Développement d'une méthode pour le problème de tournées de véhicules multi-dépôts

Réalisé par :

- Haddi Hayette
- Ameur Nesrine

Présentée et soutenue le 26 septembre 2022 devant le jury composé :

Dr. Aroussi Sana, Président. Université de Blida 1

Dr. Boudraa, Examinateur. Université de Blida 1

Dr. Ferdi Imene, Encadreur. Université Blida 1

ANNEE UNIVERSITAIRE 2021/2022

Remerciements

Nous remercions DIEU tout puissant de nos avoir donné la force, la santé, le courage et la patience de pouvoir accomplir ce travail.

Un grand merci à toutes nos familles surtout nos parents pour leur encouragement et leur suivi avec patience du déroulement de notre projet.

Nos remerciements s'étendent à notre promoteur DR Imene Ferdi pour le grand soutient morale qu'il nous a apporté au cours de notre projet et aussi pour sa patience, sa disponibilité et surtout pour ces judicieux conseils, qui ont contribué à alimenter nos réflexions.

Nos remerciements très sincères, vont aussi à tous les membres du personnel du département informatique, et en particulier, à tous ceux qui nous ont enseigné durant toutes nos années d'études universitaires. Qu'il trouve ici l'assurance de notre profonde gratitude.

Enfin, nous remercions tous nos amis avec qui nous partagé des moments d'entraides, de faiblesse, de souffrance et de joie, et aussi à tous ceux qui ont contribué de près ou de loin à l'achèvement de ce travail.

MERCI !

DEDICACE

Nous dédions cet humble travail

A nos chers parents qui nous ont donné un magnifique modèle de labeur et de persévérance.

A nos frères et sœurs à qui nous souhaitons un avenir prospère.

A toute nos familles sans exception.

A tous nos amis et collègues.

A tous ceux qui nous aidés.

A tout le personnel du département informatique.

A tous ceux que nous avons omis.

Hayette & Nesrine

Résumé

Le domaine de l'optimisation combinatoire est fortement présent dans de nombreux secteurs de recherche et d'industrie. Parmi les secteurs, où on trouve plus de problèmes d'optimisation : les secteurs de transport et planification de production. En effet, plusieurs problèmes sont issus de ces deux secteurs comme le problème de localisation des usines, le problème de tournées de véhicules…etc.

Le problème de tournées de véhicules (VRP) est l'un des problèmes les plus connus dans la recherche opérationnelle. Dans le cadre de ce travail de recherche, nous sommes intéressés d'étudier le problème de tournées de véhicules multi-dépôts (Multi-Depots Vehicule Routing Problem MDVRP) qui est une extension de VRP classique. Ce problème est NP-difficiles permettant de déterminer les tournées de véhicules associées à chaque dépôt pour satisfaire un ensemble de clients, tout en minimisant le coût total de la livraison.

La motivation de ce travail est d'aborder un problème difficile et dans une version encore peu étudiée dans la littérature, comportant des capacités limitées à la fois pour les dépôts et les véhicules réalisant les tournées. De plus, des problèmes de taille réaliste sont visés.

L'objectif de ce mémoire est de développer une méthode efficace, simple et robuste pour la résolution de MDVRP. Dans le but de réaliser notre objectif, nous avons proposé une adaptation de l'algorithme inspiré des réactions chimiques (CRO) pour ce problème.

Afin de tester la performance de l'approche que nous avons proposée, nous l'avons testée et validée sur un ensemble de tests provenant de la littérature. Les études expérimentales faites ont montré l'efficacité de la méthode proposée de trouver des solutions de bonne qualité pour le problème étudiée dans ce mémoire.

Mots-clés : Méthodes d'optimisation combinatoire, Problème de tournées de véhicules, Problème de tournées de véhicules multi-dépôts Méta-heuristiques, Algorithme inspiré des réactions chimiques (CRO).

Abstract

The field of combinatorial optimization is strongly present in many research and industry sectors. Among the sectors, where there are more optimization problems: the transport and production planning sectors. Indeed, several problems arise from these two sectors such as the problem of locating factories, the problem of vehicle routing, etc.

The vehicle routing problem (VRP) is one of the well-known problems in operations research. As part of this research work, we are interested in studying the multi-depots vehicle routing problem (Multi-Depot Vehicle Routing Problem MDVRP) which is an extension of classic VRP. This problem is NP-hard, the objective is to determine the vehicle routes associated to each depot to satisfy a set of customers, while minimizing the total cost of delivery.

The motivation of this work is to approach a difficult problem and in a version still little studied in the literature, comprising limited capacities at the same time for the depots and the vehicles. In addition, realistic size problems are targeted.

The objective of this work is to develop an efficient, simple and robust method for solving MDVRP. In order to achieve our goal, we proposed an adaptation the chemical reaction optimization algorithm (CRO) for this problem.

In order to test the performance of the proposed approach, we tested and validated it on a set of benchmarks from the literature. Experimental studies have shown the effectiveness of the proposed method to finding good quality of solutions for the problem studied in this work.

Keywords: Combinatorial optimization methods, Vehicle routing problem, Multi-depot vehicle routing problem, Meta-heuristics, chemical reaction optimization algorithm (CRO).

ملخص

مجال التحسين الاندماجي موجود بقوة في العديد من قطاعات البحث والصناعة. من بين هذه القطاعات نجد قطاعا النقل وتخطيط الإنتاج. في الواقع، تنشأ العديد من المشاكل من هذين القطاعين مثل مشكلة تحديد مواقع المصانع، مشكلة توجيه المركبات، إلخ.

تعد مشكلة توجيه المركبات واحدة من أكثر المشكلات المعروفة في أبحاث العمليات. كجزء من هذا العمل البحثي، نحن مهتمون بدراسة مشكلة توجيه المركبات متعددة المستودعات والتي تعد حالة خاصة لمشكلة توجيه المركبات الكلاسيكي. هذه المشكلة هي مشكلة صعبة الهدف منها تحديد مسارات المركبات المرتبطة بكل مستودع لإرضاء مجموعة من العملاء، مع تقليل التكلفة الإجمالية للتسليم.

الدافع من هذا العمل هو التعامل مع مشكلة صعبة وفي نسخة لا تزال تدرس قليلاً في الأدبيات، تشتمل على قدرات محدودة في نفس الوقت للمستودعات والمركبات التي تقوم بالجولات. بالإضافة إلى ذلك، يتم استهداف مشاكل الحجم الواقعي.

الهدف من هذه الأطروحة هو تطوير طريقة فعالة وبسيطة وقوية لحل مشكلة توجيه المركبات متعددة المستودعات. من أجل تحقيق هدفنا، اقترحنا تكييف الخوارزمية المستوحاة من ردود التفاعلات الكيميائية لهذه المشكلة.

من أجل اختبار أداء المنهج الذي اقترحناه، قمنا باختباره والتحقق من صحته في مجموعة من الاختبارات من الأدبيات. أظهرت الدراسات التجريبية فاعلية الطريقة المقترحة في إيجاد حلول ذات نوعية جيدة للمشكلة التي تمت دراستها في هذه الرسالة

الكلمات المفتاحية: طرق التحسين التجميعية، مشكلة توجيه المركبات، مشكلة توجيه المركبات متعددة المستودعات، الاستدلال الفوقي، الخوارزمية المستوحاة من ردود التفاعلات الكيميائية.

Table des matières

Liste des figures

Liste des Tableaux

Liste des abréviations

ACO : Ant Colonie Optimization

AG : Algorithme Génétique

B & B : Branch and Bound

BKS : Best Known Solution

BVDH : Two-Level Voronoi Diagram

CRO : Chemical Reaction Optimization

CVRP : Capacited Vehicule Routing Problem

DALO : Discrete Antlion Optimization

DVRP : Dynamic Vehicule Routing Problem

FIFO : First In First Out

GTS : Granular Tabu Search

GVNS : General Variable Neighborhood Search

HVRP : Heterogeneous Vehicule Routing Problem

KE : Energie Cinétique

MDVRP : Multi Depot Vehicule Routing Problem

MDGVRP : Multi depot Green Vehicule Routing Problem

MHDT : Merge-Head and Drop-Tail

MPVRP : Multi-Periodic Vehicule Routin Problem

MPVRP : Multipe Product Vehicule Routing Problem

OVRP : Open Vehicule Routing Problem

PD : Programmation Dynamique

PE : Energie Potentielle

PL : Programmation Linéaire

PLNE : Programmation Linéaire en Nombre Entier

PSO : Particle Swarm Optimization

RL : Recherche Locale

RS : Recuit Simulé

TOP : Team Orenteering Problem

TS : Tabu Search

TSP : Traveling Saleseman Problem

VNS : Variable Neighborhood Search

VRP : Vehicule Routing Problem

VRPB : Vehicule Routing Problem with Backhauls

VRPCB : Vehicule Routing Problem with Clusteres Backhauls

VRPFL : Vehicule Routing Problem with Full Load

VRPHF : Vehicule Routing Problem With Heterogeneous Fleet

VRPMB : Vehicule Routing Problem With Mixed Linehauls and Backhauls

VRPTW : Vehicule Routing Problem with Time Windows

Introduction générale

Dans la vie moderne, le transport occupe une place importante, La résolution de problème de transport a toujours été la clé des problèmes de logistiques, Cette dernière est devenue une composante essentielle pour le bon fonctionnement des entreprises. L'aspect compétitif rend la recherche de solutions logistique efficaces et crucial, ces solutions tendent à réduire le coût de transport, le nombre de véhicules utilisées, les délais de livraisons, la capacité de véhicules…etc.

L'optimisation de transport et la performance logistique sont des enjeux économiques très importants pour les entreprises. Cela repose notamment sur l'organisation et l'efficacité des tournées de véhicules.

Résoudre un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Les problèmes d'optimisation sont souvent faciles à définir mais généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possède pas à ce jour de solution algorithmique efficace et valable pour toutes les instances. De nombreuses méthodes ont été développées pour résoudre ces problèmes. Ces méthodes peuvent être classées en deux catégories : les méthodes exactes et les méthodes approchées (approximatives).

Les méthodes exactes permettant de trouver la solution optimale d'un problème, mais elle a un inconvénient majeur qui est le temps d'exécution qui devient déraisonnable. Tandis que les méthodes approchées permettant de trouver des bonnes solutions, mais ne garantissent en aucun cas l'optimalité. Par conséquence, les méthodes exactes sont plus pratique pour la résolution des problèmes faciles, en revanche, les méthodes approchées sont plus pratique dans le cas où on cherche une solution de bonne qualité dans un bref délai.

Dans ce travail, nous nous sommes intéressés au domaine de la logistique du transport qui constitue un élément très important au sein des entreprises, et qui contient plusieurs problèmes complexes de nature combinatoire tels que le Problème de Tournées de Véhicules Multi-Dépôts PTVMD (MDVRP : Multi-Depot Vehicule Routing Problem) qui est un cas particulier de problème générale de tournées de véhicules (VRP). Ce dernier problème consiste à considère plusieurs dépôts au lieu d'un (dans le cas de VRP), pour chacun d'entre eux une flotte de véhicules est affecté. Chaque client doit être visité une seule fois à partir d'un seul dépôt. Les véhicules doivent commencer et finir leurs tournées au dépôt ou ils sont affectées et ne peuvent pas visiter les autres dépôts, dans le but de minimiser le coût total de la livraison.

Dans notre modélisation du problème MDVRP, la minimisation des coûts de transport, tout en respectant les contraintes de capacité, est un facteur très important. Par conséquent, le présent mémoire servira à étudier comment minimiser le coût total de trajet, alors que les clients et les dépôts sont diffusés et situés dans différents endroit.

L'objectif principal de ce projet consiste à développer et implémenter une méthode permettant la résolution du problème MDVRP, en se basant sur les approches proposées dans le domaine de l'optimisation combinatoire. Pour cela, nous avons proposé une adaptation de

l'algorithme inspiré des réactions chimiques (CRO). CRO est une méta-heuristique récente basée population. Il s'inspire du phénomène de réaction chimique et se basé sur la distribution de l'énergie entre les molécules.

La motivation derrière cette contribution est que les méta-heuristiques basées population sont généralement plus performantes que les méta-heuristiques basées solution unique.

Structuration du document

Le mémoire présenté est divisé en deux parties chacun contient deux chapitres. La première partie présente les concepts de base et l'état de l'art et la seconde partie contient notre contribution.

La première partie est composée de deux chapitres : Le chapitre 1 est dédié aux différents concepts de bases liées à l'optimisation, la complexité algorithmique, les classes de complexité et les méthodes de résolution des problèmes d'optimisations.

Le chapitre 2 s'intéresse au problème du tournées de véhicules multi-dépôts, nous commençons par la présentation du problème de tournées de véhicules classique, et identifier ses variantes. Ensuite, nous présentons le problème de tournées de véhicules multi-dépôts en abordant sa définition mathématique et les différentes méthodes de résolution développées dans la littérature pour le résoudre.

La deuxième partie décrit en détail notre contribution réalisée dans le cadre de ce mémoire. Elle comporte deux chapitres : Le chapitre 3 qui contient notre première contribution, il couvre le principe de fonctionnement de l'algorithme inspiré des réactions chimiques et ses principales caractéristiques. Nous expliquerons comment cet algorithme est appliqué au MDVRP et l'adaptation de ses opérateurs à ce problème.

Le chapitre 4 présente l'étude expérimentale. Dans ce chapitre nous présentons l'environnement de développement ainsi que le langage de programmation utilisé. Nous présentons par la suite les données de tests utilisées (Benchmarks), les résultats obtenus lors de l'application de notre approche et enfin la discussion de ces résultats.

Finalement, nous terminons avec une conclusion générale sur la contribution proposée et quelque respectives pour nos futures travaux de recherche.

Partie 1
Concepts de base et état de l’art

Chapitre 1

Introduction à l'optimisation combinatoire

1.1 Introduction

Dans la vie, les entreprises sont confrontées à de nombreuses problématiques qui deviennent de plus en plus complexes au fil du temps dans différents domaines comme le transport, l'industrie, l'économie, l'énergie, la distribution des marchandises, la production…etc. La recherche d'une solution optimale ou sous optimale dans un temps raisonnable est devenue une priorité importante dans chaque entreprise.

L'optimisation combinatoire et la prise de décision occupent en particulier une place importante pour résoudre ces nombreux problèmes. Dans le processus de décision nous devons répondre "oui "ou "non" pour savoir s'il existe une solution à ce problème, par contre, l'optimisation combinatoire cherche à optimiser la valeur d'une fonction. Généralement, Les problèmes d'optimisation combinatoire sont souvent faciles à définir et difficiles à résoudre.

Ce chapitre est une introduction à l'optimisation, en particulier les problèmes d'optimisation combinatoire. Nous nous focalisons sur la théorie de la complexité, les principales classes de complexité. De plus, la formulation mathématique des problèmes d'optimisation combinatoire et les différentes méthodes de résolution les plus connues seront présentées.

1.2 Notions de base

L'optimisation est une branche des mathématiques et de l'informatique en tant que discipline qui vise à modéliser, analyser et résoudre des problèmes pratiques de manière analytique ou numérique :

Il s'agit notamment de déterminer quelles solutions (inconnues) satisfont les objectifs quantitatifs, tout en respectant les éventuelles contraintes. Cela revient donc à résoudre un problème d'optimisation.

Les problèmes d'optimisation est un modèle mathématique (formel) d'un problème réel, on cherche à minimiser (ou maximiser) une fonction objectif (coût) f(s) par un ensemble de solutions S et un ensemble de contraintes C. Parmi les solutions S, un sous-ensemble $X \subseteq S$ représente des solutions réalisables respectant les contraintes C du problème, à chaque solution s est associée une valeur f(s) qui représente sa qualité. Résoudre un problème d'optimisation consiste à trouver une solution $s^* \in X$ qui minimise ou maximise la valeur f(s).

Définition 1.1: Conformément au [Papadimitriou et Steiglitz, 1982] : Une instance d'un problème de minimisation ou (maximisation) est un couple (X, f). Où $X \subseteq S$ est un ensemble fini de solutions possibles et f est une fonction objectif (fonction du coût) à minimiser (à maximiser), f : X → IR. L'objectif est de trouver $s^* \in X$ tel que :

$$\begin{cases} f(s*) \leq f(s), \forall s \in X & \text{(cas d'une minimisation)} \\ f(s*) \geq f(s), \forall s \in X & \text{(cas d'une maximisation)} \end{cases}$$

Définition 1.2: le problème d'optimisation est mono-objectif lorsqu'un seul objectif (critère) est donné inspirée de [Papadimitriou et Steiglitz, 1982]. Dans ce cas la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal, maximal)**.**

De manière formelle, à chaque instance d'un tel problème est associé un ensemble W des solutions potentielles respectant certaines contraintes et une fonction d'objectif W ϵ Y qui associe à chaque solution admissible s de W une valeur (s). Résoudre l'instance (W) du problème d'optimisation consiste à trouver la solution optimale s* de W qui optimise (minimise ou maximise) la valeur de la fonction objective pour le cas de la minimisation : le but est de trouver s* de W tel que (s*) ϵ (s) pour tout élément s de W. Un problème de maximisation peut être défini de manière similaire**.**

Définition 1.3: Un problème d'optimisation multi-objectif MOP de [Fonseca et Fleming, 1993] (multi-objective problem) peut être défini comme suit :

$$(\mathrm{MOP}) = \begin{cases} \mathrm{Min}\ f(x) = f1(x), f2(x), \dots\dots\dots, fn(x). \\ \text{Tel que : } x \epsilon X. \end{cases}$$

Où n est le nombre d'objectifs ($n \geq 2$), X est l'ensemble des solutions réalisable dans l'espace décisionnel, et $x = (x_1, x_2, \dots\dots, x_k) \in X$ est un vecteur représentant les variables de décision, dans le cas combinatoire x est un ensemble discret, à chaque solution $x \in X$ est associé un vecteur objectif $z \in Z$ sur la base d'un vecteur de fonction $f : X \rightarrow Z$ tel que $z = (z_1, z_2, \dots) = f(x) = (f_1(x), f_2(x), \dots. f_n(x))$, ou $Z = f(X)$ représente l'ensemble des points réalisable**.**

Définition 1.4: Une contrainte d'un problème est une restriction imposée par la nature et les caractéristiques du problème sur les solutions proposées.

Définition 1.5: L'espace de recherche S d'un problème est composé de l'ensemble de valeurs pouvant être prises par les variables ($s_1, s_2, s_3 \dots, s_n$) qui construisent la solution s.

Définition 1.6: Une solution voisine d'une solution s est définit comme une légère modification de la solution s. Cette modification est appelée aussi mouvement. Le voisinage V(s) d'une solution s est un sous ensemble de S [Devarenne, 2007].

Définition 1.7: Un optimum local reflète la meilleure solution dans un entourage voisin (voisinage V(s)) de la solution s. on dit que la solution s´ (appartenant à S) est un optimum local du voisinage V(s) de la solution s si vérifie la condition suivante :

- $f(s') \leq f(s)\ \forall s \in V(s)$ Pour un problème de minimisation.
- $f(s') \geq f(s)\ \forall s \in V(s)$ Pour un problème de maximisation.

Définition 1.8: Un optimum global (solution optimale) est Une solution $s* \in S$ est dite optimale, si elle a une meilleure valeur pour la fonction objectif f alors n'existe pas d'autres solutions de meilleure qualité c'est-à-dire le meilleur optimum local. On peut avoir plusieurs solutions globales mais un seul optimum global. [Web1]

Formellement, une solution $s* \in S$ est un optimum global, si et seulement si :

- $s^* = \arg\min\{ f(s^*) \} := \{ s^* | f(s^*) \leq f(s), \forall s \in S \}$ (cas de problème de minimisation).
- $s^* = \arg\max\{ f(s^*) \} := \{ s^* | f(s^*) \geq f(s), \forall s \in S \}$ ((cas de problème de maximisation).

La figure 1.1 représente les optimums locaux et les optimums globaux d'une fonction d'évaluation :

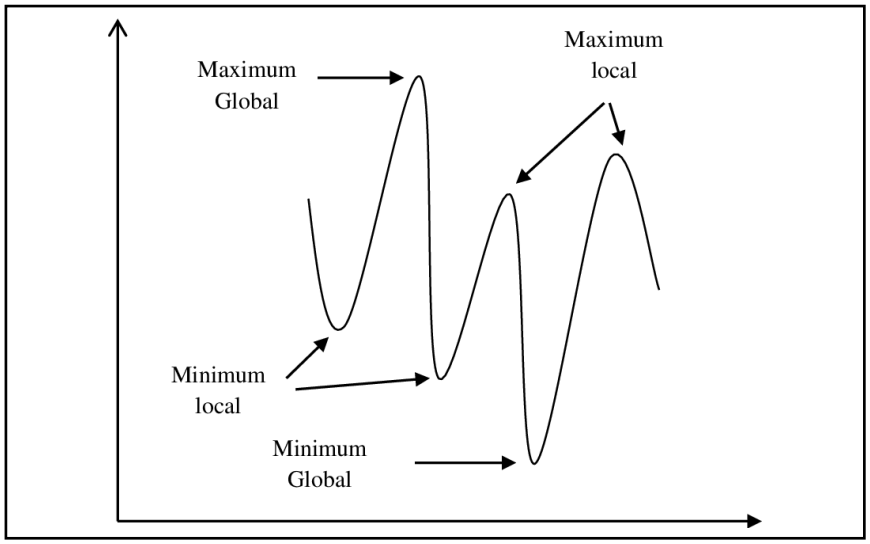

Figure 1.1. Courbe représentant les optimums locaux et les optimums globaux [Devarenne, 2007].

1.3 La théorie de la complexité

La théorie de la complexité consiste à estimer la difficulté ou la complexité d'une solution algorithmique d'un problème posé de façon mathématique [Papadimitrou, 1994]. Elle se concentre sur les problèmes de décision qui posent la question de l'existence d'une solution comme le problème de satisfiabilité booléenne [Garey et Johnson, 1979].

La complexité de tout problème consiste à estimer le nombre d'instructions à exécuter pour résoudre l'instance du problème dans le pire des cas. La complexité algorithmique sert à mesurer l'efficacité d'un algorithme de résoudre d'un problème. En effet, plus le nombre d'instructions est important plus le problème est complexe et donc la complexité algorithmique est proportionnelle au nombre d'instructions du problème. On peut déduire que plus l'algorithme est complexe moins il est efficace. Plusieurs types de complexité peuvent être distingués, on cite la complexité constante o(1) qui est indépendante de la taille du problème, la complexité logarithmique o(log(n)), la complexité linéaire o(n), la complexité quadratique $o(n^2)$, la complexité cubique $o(n^3)$, la complexité polynomiale $o(n^p)$, la complexité exponentielle o(exp n), et enfin la complexité factorielle o(n!).

En effet, il existe différentes classes de problèmes en fonction de la complexité, parmi les classes les plus courantes, on distingue :

- **La classe P** : La classe P (Polynomial time)

La classe P regroupe tous les problèmes de décision qui peuvent être résolus par un algorithme déterministe de complexité polynomiale. En d'autres termes, un algorithme permettant de trouver une solution optimale pour toutes les instances du problème en un temps polynomial par rapport à la taille de l'instance par une machine de Turing [Sakarovitch, 1984].

- **La classe NP** : La classe NP (Non-déterministe Polynomial)

Regroupe tous les problèmes de décision polynomiaux non déterministes qui peuvent être résolus par un algorithme non-déterministe de complexité polynomiale (i.e. dont la solution peut être vérifiée en temps polynomial).
Pour montrer qu'un problème est dans la classe NP, il suffit de trouver un algorithme qui vérifie si une solution donnée est valide en temps polynomiale [Web2].

- **Classe NP-complet :**

Elle contient les problèmes les plus difficiles de la classe NP, et la classe NP-Complet regroupe les problèmes de décision pour lesquels il n'existe pas d'algorithme permettant leur résolution en un temps polynomial.
Un problème de décision n est NP-complet s'il satisfait les deux conditions suivantes : n ∈ NP, et tout problème NP se réduit à n en temps polynomial [Garey et Johnson, 1979].
Il est important de préciser que tous les problèmes d'optimisation ne peuvent pas être classifiés comme problèmes NP-complets, puisqu'ils ne sont pas tous des problèmes de décision, même si pour chaque problème d'optimisation on peut définir un problème de décision qui a une complexité équivalente.

- **NP-difficile :**

Elle regroupe l'ensemble des problèmes dont la complexité est exponentielle ou doublement exponentielle. Cette classe de problème se distingue par la difficulté de résolution optimale ou exacte des grandes instances, en un temps polynomial. Les problèmes NP-Difficiles sont aussi difficiles que les problèmes NP Complet, on peut observer que pour montrer qu'un problème d'optimisation est NP-difficile, il suffit de montrer que le problème de décision associé à lui est NP-complet. Cette classe représente des problèmes d'optimisation du monde réel, il a été prouvé qu'un grand nombre de problèmes d'optimisation sont NP-difficiles. C'est notamment le cas des problèmes du Voyageur de Commerce, de Partitionnement de Graphes et d'Affectation Quadratique...etc.
Le problème de tournées de véhicules est l'un des problèmes de routage. Les auteurs Lenstra, Rinnooy et Kan en 1988 [Lenstra et Kan, 1988] ont prouvé que le VRP est un problème NP-difficiles. La figure 1.2 montre La relation entre les différentes classes [Web3].

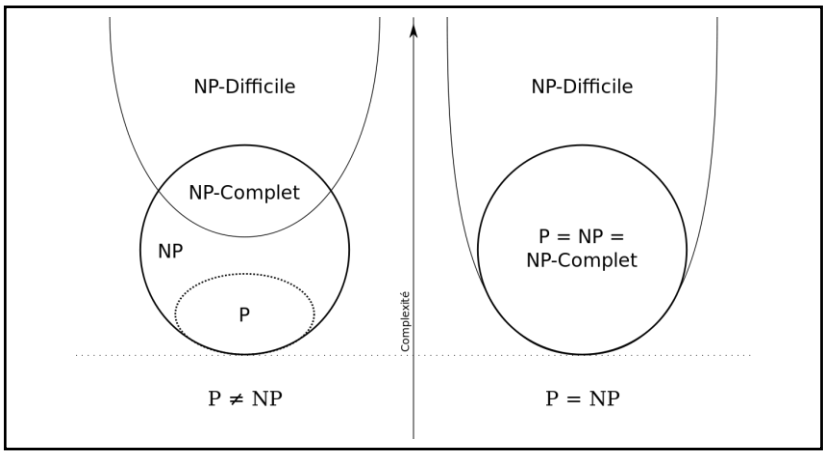

Figure 1.2. La relation entre les classes P, NP, NP-complet, NP-difficile [Web3].

1.4 Les problèmes d'optimisation combinatoires

L'optimisation combinatoire occupe une place très importante en informatique et en mathématique, et vise à trouver la meilleure solution, d'un problème donné, parmi un ensemble d'alternatives possibles [Papadimitriou et Steiglitz, 1982]. De nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation dans différents secteurs : Télécommunications, Électronique, Mécanique, Chimie, biologie, Transport...etc.

Les problèmes d'optimisation combinatoire sont des problèmes d'optimisation dont les solutions réalisables sont dénombrables mais de taille exponentielle consiste à-il Trouver une solution optimale à partir d'un ensemble fini et discret de solutions, ou plus formellement : minimiser ou maximiser une fonction sur un ensemble fini et discret (potentiellement très grand) [Sakarovitch, 1984], il peut être défini par un quadruplet (S, f, C, s*) où :

- S : représente l'espace de recherche ou l'ensemble de solutions possibles.
- $X \subseteq S$: est l'ensemble des solutions réalisables.
- f : $X \rightarrow$R, est la fonction « objectif » à optimiser permet de définir une relation d'ordre entre chaque pair de solutions.
- C : L'ensemble de contraintes qui définit des conditions sur l'espace d'états que les variables doivent satisfaire, permettent en général de limiter l'espace de recherche (est l'ensemble des solutions réalisables).
- s*: l'optimum global :
  - Minimum global ou le maximum global de la fonction f
  - s* vérifie l'inégalité suivante :
    - Dans le cas de minimisation : $\forall s \in X , f(s*) \leq f(s)$
    - Dans le cas de maximisation : $\forall s \in X , f(s*) \geq f(s)$

Ces problèmes sont souvent faciles à définir, mais ils sont généralement difficiles à résoudre. Ils appartiennent à la classe des problèmes NP-difficiles, et ne possèdent pas à ce jour de solution algorithmique efficace valable pour toutes les données.

Il existe plusieurs techniques pour résoudre les problèmes d'optimisation, classées en deux grandes classes : les méthodes exactes qui donnent l'optimalité de la solution et les méthodes approchées qui donnent des solutions acceptables en temps raisonnable (pas optimale).

1.5 Classification des problèmes d'optimisation

Face à la résolution d'un problème d'optimisation, il est important de bien identifier à quelle catégorie ce problème appartient. En effet, les algorithmes développés sont conçus pour résoudre un type de problème donné et sont peu efficaces pour un type différent. La classification des problèmes d'optimisation change d'un auteur à un autre. Par exemple [Xin, 2010], on distingue :

- **Classifier selon le type des variables :**

Dans certains cas, les variables de décision sont discrètes, le plus souvent sous la forme d'entiers ou de binaires, le problème d'optimisation est dit discret (l'optimisation discrète est également appelée optimisation combinatoire). En revanche, dans les problèmes d'optimisation continue, les variables peuvent prendre n'importe quelle valeur, ce sont des nombres réels (les variables). Les problèmes d'optimisation continue sont généralement plus simples à résoudre. Un problème d'optimisation est dit mixte si ces variables peuvent être à la fois discrètes et continues.

- **Classifier selon le nombre de contraintes :**

Il existe deux types de problèmes : les problèmes d'optimisation avec et sans contrainte, Il est important de bien distinguer les problèmes où des contraintes existent sur les variables de décision. Ces contraintes peuvent être simplement des bornes et aller jusqu'à un ensemble d'équations de type égalité et de type inégalité. Il est parfois possible d'éliminer une contrainte égalité par substitution dans la fonction objectif. Généralement, les problèmes avec contraintes sont plus compliqués à résoudre et utilisent des algorithmes dédiés.

- **Classifier selon le nombre d'objectifs :**

C'est des problèmes d'optimisation mono-objectif ou multi-objectif, les problèmes mono-objectifs sont définis par une unique fonction objectif. Le problème des objectifs multiples (multi-objectifs) se pose lorsqu'il s'agit de trouver un compromis entre plusieurs objectifs contradictoires. Il est éventuellement possible (mais pas nécessairement efficace) de reformuler un problème multi-objectif avec une seule fonction objectif sous forme d'une combinaison des différents objectifs ou en transformant des objectifs sous forme de contraintes. Il est important de noter que la plupart des problèmes d'optimisation dans le monde réel sont multi-objectifs.

- **Classifier selon les valeurs des variables de décision et de la fonction objectif :**

Elles sont définies sur des problèmes d'optimisation déterministe ou stochastique (dynamique), les problèmes d'optimisation déterministe considèrent que les données qui sont connues parfaitement, alors que dans les problèmes d'optimisation stochastique (dynamique), ce n'est pas le cas. Par exemple, une approche stochastique peut être pertinente dans l'état où les variables d'un problème sont les ventes futures d'un produit. Dans cette condition, l'incertitude peut être introduite dans le modèle.

❖ **Classifier selon le paysage de la fonction objectif :**

On a deux types de problèmes d'optimisation unimodal et multimodal :
Le problème d'optimisation unimodal utilise les méthodes d'optimisation unimodale cherchent à trouver une solution globalement optimale pour un problème donné, le problème d'optimisation multimodal utilise les méthodes d'optimisation multimodale cherchent à trouver un ensemble de bonnes solutions.

❖ **Classifier selon la nature des fonctions du problème :**

On dit que les problèmes d'optimisation sont linéaires si la fonction objectif et les contraintes sont linéaires, et pour les problèmes d'optimisation non linéaires si la fonction objectif et les contraintes sont représentées par des relations non linéaires. La figure 1.3 montre une classification pour les problèmes d'optimisation.

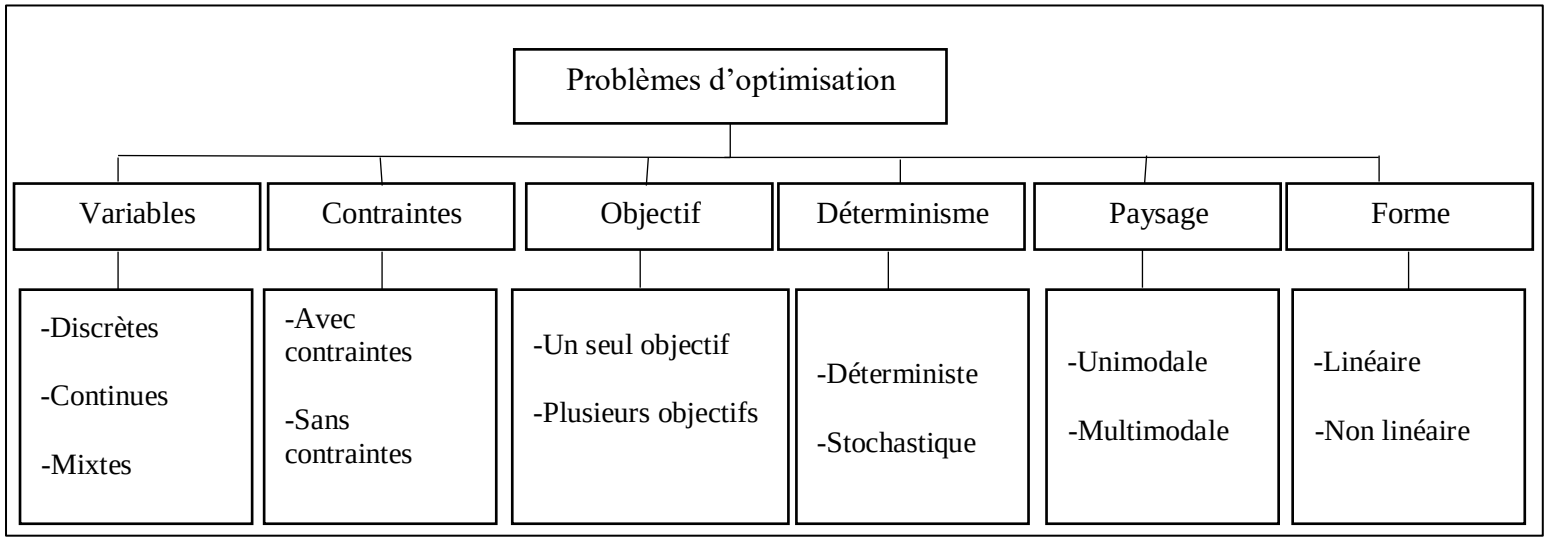

Figure 1.3. Classification des problèmes d'optimisation [Xin, 2010].

1.6 Les méthodes de résolution des problèmes d'optimisation

Les problèmes d'optimisation combinatoire peuvent s'avérer très difficiles à résoudre bien qu'ils soient généralement faciles à formaliser. Par conséquence, de nombreuses méthodes de résolution de différents problèmes ont été proposés dans la littérature. Selon EL Ghazali, [Talbi, 2009], il existe deux classifications qui se concentrent sur les objectifs et l'exactitude de la solution. Dans ce mémoire, nous nous intéressons principalement aux méthodes de résolution de la deuxième classification (voir la figure 1.4) ou ces méthodes se regroupent en deux grands familles la première englobe les méthodes dites exactes et la deuxième famille dites approchées.

Le schéma ci-dessous représente cette classification:

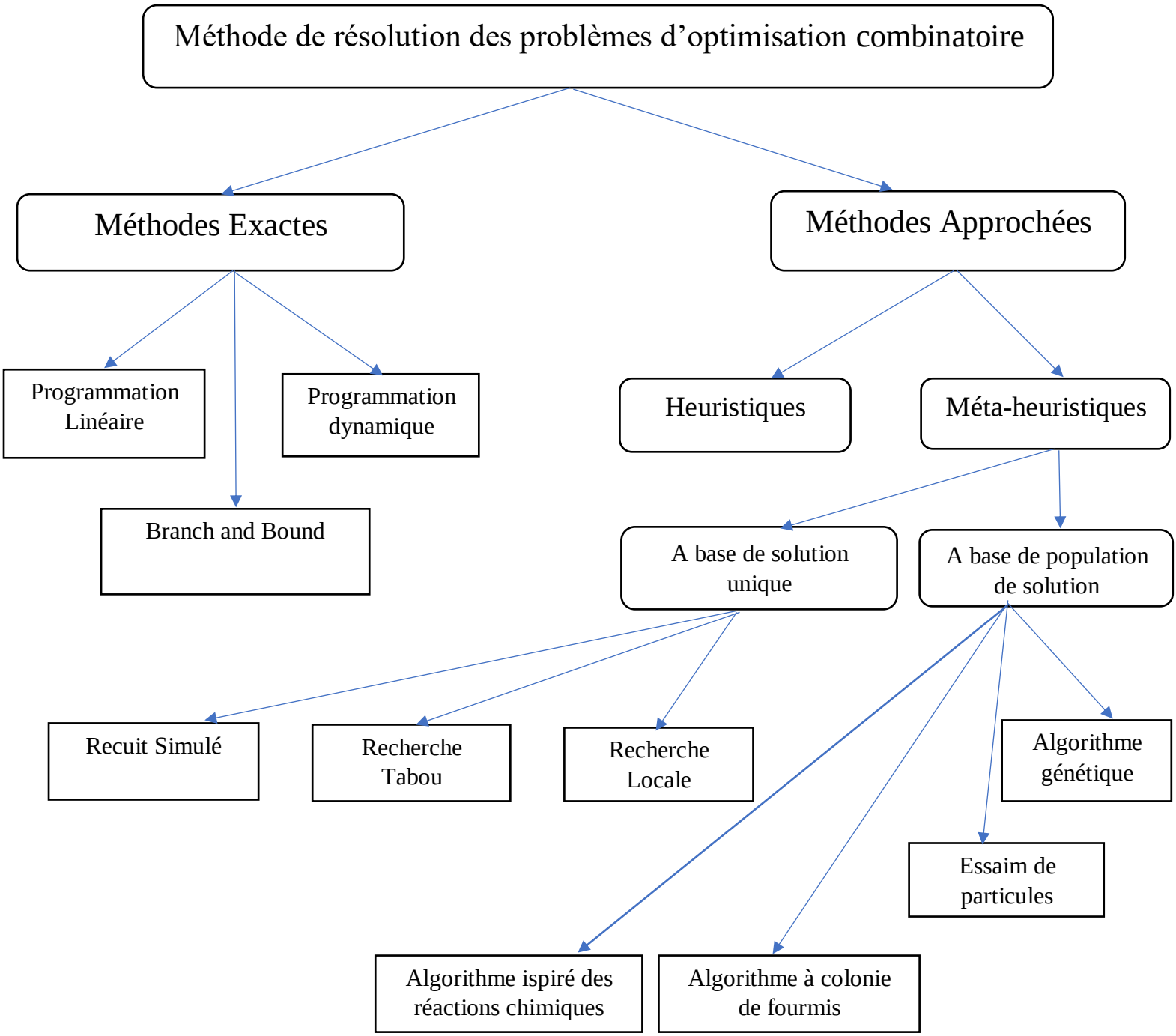

Figure 1.4. Classification de méthodes de résolution de problèmes d'optimisation. [Talbi, 2009]

1.6.1 Les méthodes exactes

Le principe des méthodes exactes est de trouver souvent implicitement une solution, une solution optimale ou un ensemble de solution à un problème. Ce type de méthode permettant de trouver une solution optimale d'un problème et de prouver l'optimalité, sur les problèmes issus d'applications spatiales. Elles sont efficaces en pratique sur des instances de petite taille, mais malheureusement, ces méthodes exigent des coûts de résolution en termes de temps de calcul et d'espace mémoire souvent prohibitifs et qui augment exponentiellement avec la taille du problème traité.

Il existe de nombreux algorithmes exacts, y compris programmation dynamique, algorithme simplexe, algorithme de séparation et d'évaluation (Branch & bound), algorithme de Backtracking…etc. Notre mission n'est plus de relier des principes disparates les méthodes exactes, mais référencez-en quelques-unes ci-dessous.

1.6.1.1 La programmation dynamique

La programmation dynamique (PD) a été introduite au début des années 1950 par Richard Bellman [Bellman, 1950], à l'époque « programmation » signifie planification et ordonnancement. Le but de cette méthode est de résoudre des problèmes d'optimisations pour déduire une solution optimale. Cette méthode est applicable à tout type de problème en le décomposant en sous-problèmes, puis résoudre ces sous-problèmes de manière ascendante, c'est-à-dire qu'on débute de plus petite aux plus grands pour ensuite déduire progressivement les solutions de l'ensemble. Autrement dit : la programmation dynamique est une technique d'optimisation visant à éviter de recalculer des sous-problèmes en stockant les résultats en mémoires.

La programmation dynamique est très efficace pour la résolution des problèmes de petite et moyenne tailles.

1.6.1.2 La programmation linéaire

La programmation linéaire (PL) est une méthode exacte de l'optimisation dont la fonction objectif et les différentes contraintes sont linéaires.

La programmation linéaire est simple à résoudre numériquement, cette méthode est essentiellement appliquée pour résoudre des problèmes d'optimisations à moyen et long terme. Beaucoup de problèmes de recherche opérationnelle peuvent être exprimés comme des problèmes d'optimisation linéaire. Pour cette raison, un grand nombre d'algorithme pour la résolution d'autres problèmes d'optimisation sont fondées sur la résolution de problème linéaire.

Les solutions trouvées en termes de programmation linéaire doivent être représentés en variables réelles, et quand on parle à la programmation linéaire en nombre entier (PLNE) il est nécessaire d'utiliser des variables discrètes. On trouve beaucoup des algorithmes qui permettent de résoudre le problème de PL comme l'algorithme de simplexe.

1.6.1.3 L'algorithme Branch and Bound

Le mot Branch signifie : Branchement (séparation) et bound signifie : borne (calcul de la borne ou évaluation de nœud). La méthode a été proposée pour la première fois par Alisa Land et Alison Doig [Alisa et Alison, 1960] lors de recherches à London School of Economics parrainée par British Pelroleum en 1960 pour la programmation discrète, et est devenue l'outil le plus couramment utilisé pour résoudre les problèmes d'optimisation de types NP-difficiles.

Les méthodes B&B sont des méthodes basées sur une énumération ''intelligente'' des solutions admissibles d'un problème d'optimisations combinatoires, il s'agit d'une méthode d'optimisation exactes pour résoudre les problèmes d'optimisations au moins NP-difficiles ou NP-complets au sens fort. Le principe de base de cette méthode vient du comportement : « Diviser pour régner ». Elle consiste à dissocier le problème en sous-ensembles de manière à représenter le problème sous forme d'une arborescence.

L'algorithme branch and Bound possède deux phases : la séparation et l'évaluation, la phase de séparation consiste à séparer un ensemble de problèmes en sous-ensembles, plus simple à résoudre. En pratique, on offre un « branchement » dans l'arbre, tout en garantissant la conservation de toutes les solutions admissibles du problème avant branchement. Tandis que la phase d'évaluation consiste à évaluer les solutions d'un sous-ensemble en trouvant une borne supérieure en cas de maximisation et une borne inferieure en cas de minimisation, ou au contraire de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème initial.

L'utilisation de la méthode B&B nécessite :

- Une solution initiale permettant d'entamer la recherche.
- Une stratégie permettant la division du problème P en sous problème P_i.
- Une fonction permettant le calcul de différentes bornes.
- Une stratégie de parcours de l'arbre parcourir en profondeur, en largeur…etc.

Ce type d'algorithme est attiré l'attention de beaucoup de chercheures. L'avantage principal de cette méthode est qu'il n'y a pas de répétition et la complexité temporelle moindre par rapport à d'autres algorithmes, mais cet algorithme est limité aux réseaux de petite taille.

1.6.2 Les méthodes approchées

La majorité des problèmes d'optimisation combinatoire font partie de la classe NP-difficiles la complexité des problèmes rend les méthodes de résolution exactes inefficaces, donc les chercheurs ont été menés à développer une autre type de méthode nommé méthodes approchées. Ce sont des méthodes incomplètes, le grand avantage de méthode approchée c'est qu'elles offrent des bonnes solutions très rapidement, Mais elles ne garantissent pas l'optimalité des solutions

De nombreux méthodes approchées ont été proposées, elles sont plus pratiques pour la résolution de problèmes difficiles ou de problèmes dont on cherche des solutions en un bref

délai. Ces méthodes peuvent être divisées en deux sous familles : les méthodes heuristiques et les méthodes méta-heuristiques.

1.6.2.1 Les heuristiques

Le terme heuristique vient du verbe grec ‘’heuriskein’’ signifiant trouver, une heuristique permet de trouver une bonne solution, en un temps raisonnable, mais cette solution n’est pas forcement optimale. En recherche opérationnelle, les heuristiques sont des règles empiriques simples qui ne sont pas basées sur l’analyse scientifique, elles sont basées sur l’expérience.

D’autre manière, on peut dire que les heuristiques sont des moyens pour guider le choix que doit faire un algorithme pour réduire sa complexité, elles sont spécifiques à un seul problème (n’est pas généralisé).

Selon [Feignebaum et Feldman, 1963], une heuristique est une règle d’estimation, une stratégie, une astuce ou bien une simplification, qui limite la recherche de bonnes solutions dans l’espace des configurations de façon ahurissante.

1.6.2.2 Les méta-heuristiques

Le préfixe « méta » signifie « au-delà », « plus que ça », lorsqu’il est rajouté à l’heuristique il donne méta-heuristique qui veut dire « trouver plus que ça », elles sont des heuristiques puissantes.

Les méta-heuristiques sont des méthodes d’optimisation approchées (approximatives) qui peuvent impliquer l’aspect aléatoire dans leur processus de recherche et qui permettent d’aborder des instances de problèmes difficiles en fournissant des solutions satisfaisantes (solution optimale ou proche de l’optimale) dans un délai raisonnable. On peut dire aussi, que ce sont une forme d’algorithme d’optimisation stochastique, hybridées avec une recherche locale, elles sont simples à comprendre, à concevoir et à mettre en œuvre, peuvent être combinées avec d’autres techniques d’optimisation.

Les méta-heuristiques sont des heuristiques génériques c’est-à-dire une heuristique qui peut s’appliquer pour résoudre un nombre plus large de problèmes, mais malheureusement l’inconvénient majeur de ces méthodes est qu’elles ne garantissent pas l’optimalité et ne peuvent pas les prouver.

Il existe de nombreux critères de classification des méta-heuristiques, cette classification peut être à base de solution unique ou à base de population de solution. La méta-heuristique à base de solution unique commence par une solution initiale qui sera modifiée de manières itérative, on peut citer la recherche à voisinage variable, la recherche taboue, le recuit simulé…etc. On ce qui concerne la méta-heuristique à base de population de solution, contrairement aux méthodes à solution unique, font évaluer simultanément un ensemble de solution dans l’espace. Pour ce type on distingue plusieurs méthodes, les plus connus sont : l’optimisation par colonie de fourmis, l’optimisation par essaim particulier, l’algorithme génétique, algorithme des abeilles, la recherche coucou et beaucoup d’autres qui ont été proposées ces derniers années. Dans ce qui suit nous présentons quelques méta-heuristiques dans les deux cas.

1.6.2.2.1 À base de solution unique

Une méta-heuristique à base de solution unique appelé aussi méthode trajectoires commencent avec une seul solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes trajectoires englobent essentiellement : le recuit simulé, la recherche taboue, la recherche locale, la recherche à voisinage variable(VNS)…etc.

A) Le recuit simulé

Le recuit simulé (Simulates annealing SA) est la premier méta-heuristique qui a été proposé, comme une généralisation de la méthode Monte- Carlo. En 1953, la méthode de RS est réalisée par [Metropolis et al., 1953] pour simuler l'évaluation de ce procès de recuit physiques. Après en 1983, elle a été mise au point par trois chercheurs de la société IBM, S.Kirkpatrick, C.D.Gelatt et M.P.Vecchi [Kripatrick et al., 1983] au état unis, et indépendamment par V.Cerny [Cerney, 1985] en Slovaquie à partir de l'algorithme de Métropolis, qui permet de décrire l'évolution d'un système thermodynamique, le but principal de RS est de trouver une solution optimale pour un problème donnée.

Le RS est un algorithme qui améliore l'heuristique de recherche local pour s'échapper du piège de l'extrema (minima ou maxima) local. Le recuit est un processus utilisé en métallurgie pour obtenir un alignée sans défaut. A très haut température le métal est à l'état liquide et les atomes se déplacent librement, donc on procède à un refroidissement pour revenir à l'état solide. Autrement on peut dire que le RS utilise ce qu'il est commun appelé un schéma de refroidissement où à chaque niveau de température, un certain nombre d'itération de l'algorithme est effectué.

Comme pour toute méta-heuristique, la méthode de recuit simulé est simple, facile à implémenter, donc peut être appliqué dans de nombreux problèmes d'optimisation. Les chercheures l'ont utilisée essentiellement dans : le problème de sac à dos, la segmentation d'images, le routage des paquets dans les réseaux, problème de voyageur de commerce et la conception des circuits intégrés. Par exemple le problème de voyageur de commerce le but etait de trouver le circuit hamiltonien de coût minimale dans un graphe. Par contre, le RS est un algorithme lent surtout avec les problèmes de grande taille.

B) La recherche Locale

La recherche locale simple ou la descente (RL) est un algorithme d'amélioration très ancien, c'est une méthode générale qui peut être intégré dans d'autres méthodes. Cette stratégie algorithmique est utilisée pour résoudre les problèmes d'optimisations sans contraintes.

La recherche locale consiste à passer d'une solution à une autre solution proche dans l'espace de recherche jusqu'à' ce qu'une solution considérée comme optimale soit trouvée, ou que le test d'arrêt considéré est atteint [web5]. Les algorithmes de recherche locale sont largement utilisés dans les problèmes d'optimisation difficiles, telle que : les problèmes informatique (particulier l'intelligence artificielle), les problèmes en mathématiques, les problèmes de bio-informatique, les problèmes de recherche opérationnelle, le problème de sommets, leur objectif est de trouver la solution avec le nombre de nœuds, le problème de voyageur de commerce ou TSP, le problème SAT…etc.[web 6]

L'avantage majeur de la recherche locale est sa capacité de trouver une solution acceptable (bonne qualité) dans un temps raisonnable. Le principal défaut de la méthode RL est son arrêt au premier minimum local rencontré et ne garantit pas de solution optimale.

C) La recherche Tabou

La recherche Tabou (Tabou Search TS) a été développée dans un cadre particulier par Fred Glover [Glover,1986], cette méthode se basant sur une recherche itérative qualifiée de recherche local au temps de calcul au sens large, interdit pour un laps de temps donnée de revisiter une solution déjà visité. [Web 8]

L'idée de l'approche Tabou est de rajouter un peu d'intelligente dans cette recherche pour éviter de passer deux fois au même endroit. Ainsi, Fred Glover a proposé d'introduire une mémoire qui stocke les solutions déjà testées ou de caractéristiques de celles-ci pendant une certaine durée. Ces solutions sont stockées dans une structure qui s'appelle une liste Tabou qui permet d'interdire le mouvement vers une solution voisine. On peut représenter un liste Tabou par une file d'attente de type FIFO (First In First Out).

Le principe de l'algorithme RT est l'exploitation d'autre solution en appliquant un opérateur de voisinage, on aura plusieurs solution de voisinage, donc on sélectionne la meilleure parmi ces solutions, cette solution choisie est stockée dans une liste Tabou, le temps que la solution reste dans la liste dépend de la longueur maximale de cette liste. Le principe de la méthode Tabou est : « Renoncer pour avancer ».

La recherche Tabou est une amélioration de la descente locale, elle converge vite vers une bonne solution. Cette méthode est très utilisée dans les problèmes de transport, la planification et ordonnancement, l'optimisation de graphes…etc. L'avantage de RT est son fonctionnement simple à comprendre, rapide et donne très bonne résultat sur certain type de problème.

1.6.2.2.2 À base de population

Les méta-heuristiques à base de population améliorent, au fur et à mesure dans des itérations, une population des solutions. L'idée est d'utilisé un ensemble de solution au lieu d'une seule solution pour renforce la diversité de la recherche et augmenter la possibilité d'émergence de solution de bonne qualité.

L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité. Dans cette classe on peut citer différentes méthodes telle que : les colonies de fourmis, les algorithmes de essaim de particules, l'algorithme génétique, l'algorithme des abeilles…etc.

A) L'optimisation par essaim de particules

L'optimisation par essaim de particules (particle Swarm Optimization PSO) est l'un des méta-heuristiques à base de population. Cette technique a été inventée par Russel Eberhart et James Kennedy [Kennedy et Russell, 1995], inspiré d'un comportement social d'oiseaux flottant pour résoudre les problèmes d'optimisation. Cette méthode d'optimisation basée sur la collaboration des individus entre eux comme les bancs des poissons, les oiseaux migrateurs ou les abeilles.

Dans l'espace de recherche au départ de l'algorithme, chaque particulier est caractérisé par une vitesse de déplacement et une position. Les particules sont bougées à chaque itération en fonction de trois composantes : ses vitesses actuelles, ses meilleures positions et ses meilleures positions obtenues dans le voisinage. A partir de ces informations la particule va suivre une tendance faite, d'une part, de sa volonté à retourner vers sa solution optimale.

L'algorithme PSO est utilisé comme fonction d'optimisation dans des problèmes complexe : comme gene clustering par [Xioa et al., 2003], multimodel biometric systèmes [veeramachaneni et Osadan, 2005]…etc. L'avantage majeur de PSO est la facilité de l'implémentation, un autre point fort de cette méthode est que chaque particule est soumise à des règles de déplacement très simple, qui mènent cependant l'essaim à converger rapidement vers un optimum.

B) Colonie de fourmis

L'algorithme de colonie de fourmis est l'un des algorithmes basé sur l'intelligence par essaim, initialement introduit par Marco Dorigo et ses collègues [Dorigo et al., 1991], [Dorigo, 1992], [Dorigo et al., 1996], l'idée de cette méthode est inspirée dans le comportement collectif des fourmis réelles à la recherche de nourriture. L'objectif de comportement des fourmis est de collecter la nourriture sans perdre de chemin menant à leur nid. La vie des fourmis est basée sur la collaboration entre eux, le moyen de communication entre ces fourmis s'appelle la phéromone. Lorsqu'une fourmi déplace, elle laisse une quantité variable de phéromone sur son chemin, qui est détectées par les prochaines fourmis et qui détermine avec une grande probabilité leur chemin. Donc, plus de fourmis suivent le chemin, le chemin doivent attractif.

Le fonctionnement de cette technique est comme suite : lorsque la première fourmi trouve la source de nourriture via un chemin quelconque, elle revient au nid en laissant derrière elle une piste de phéromone. Après ça les fourmis empruntent indifféremment les différents chemins possible, mais le renforcement de la piste rend plus attractif le chemin le plus court. En fin les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromone. [Web 5]

L'algorithme de colonies de fourmis est utilisée dans différents domaines d'applications comme : les applications au problème d'ordonnancement, les applications au problème de voyageur de commerce, les applications aux problèmes de tournées de véhicules, aux problèmes de coloration de graphe…etc.

C) Algorithme génétique

Les algorithmes génétiques (Genetic Algorithm GA) sont de catégorie d'algorithmes d'optimisation basée sur la recherche stochastique, appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte pour le résoudre en un temps raisonnable.

Les premiers travaux de John Holland remontent aux années 1960 et ont trouvé un premier aboutissement en 1975 avec la publication de « Adaptation in Natural and Artificial Systems » [Holland, 1975]. C'est cependant l'ouvrage de David Goldberg [Goldberg, 1989]. qui a largement contribué à développer les algorithmes génétiques.

GA s'inspire du processus de sélection naturelle qui est utilisé le même vocabulaire que celui de la biologie et la génétique classique. On parle de gêne, chromosome, individu, population et génération.

Ces mécanismes peuvent être utilisés pour spécifier un algorithme génétique par :

1. Construire une population initiale de N solutions.

2. Évaluer chacun des individus de la population.

3. Générer de nouvelles solutions en sélectionnant les parents de façon proportionnelle à leur évaluation. Appliquer les opérateurs de croisement et de mutation lors de la reproduction.

4. Lorsque N nouveaux individus ont été génères, ils remplacent alors l'ancienne population. Les individus de la nouvelle population sont évalués à leur tour.

5. Si le temps alloué n'est pas dépassé (ou le nombre de générations maximal n'est pas atteint), retourner à l'étape 3.

Pour adapter un algorithme génétique à un problème d'optimisation combinatoire, il suffit alors de spécialiser certaines composantes à la structure particulière de ce dernier. En général, les décisions portent sur les aspects suivants :

- **Codage des solutions** : il faut établir une correspondance entre les solutions du problème et les chromosomes.
- **Opérateurs génétiques** : il faut définir des opérateurs de croisement et de mutation pour les chromosomes.
- **Évaluation des chromosomes** : Dans une population où la mesure d'adaptation d'un individu est évaluée par une fonction de fitness f, l'opérateur classique consiste à associer à chaque individu une probabilité d'être sélectionné proportionnelle à la valeur de f pour cet individu. Pour choisir deux parents, on peut donc sélectionner aléatoirement 5 individus et ne garder que les deux meilleurs. Dans d'autres méthodes, les individus sont ordonnés selon leur valeur pour la fonction f, et la sélection s'effectue aléatoirement selon le rang occupé par chaque individu. On utilise alors une distribution biaisée en faveur des individus en tête de liste.

Un autre mécanisme important est le mode de remplacement des individus de la population. Dans le modèle original, chaque population de N individus est totalement remplacée à chaque génération. D'autres stratégies « élitistes » font en sorte d'assurer à chaque génération la survie des meilleurs individus de la population. C'est le cas entre autres des modèles « stationnaires » où un nombre restreint d'individus est remplacé à chaque génération.

Dans ce cas, on ajoute souvent des mécanismes supplémentaires empêchant l'ajout d'individus identiques dans la population.

L'inconvénient majeur d'AG réside dans le coût d'exécution important en comparant avec d'autres algorithmes.

1.7 Algorithme inspirées des réactions chimiques

L'algorithme inspirée des réactions chimiques CRO (anglais : Chemical Reaction Optimisation Algorithm) est une méta-heuristique basée population très intéressante. Ce dernier s'inspire du phénomène de réaction chimique qui basé sur la distribution d'énergie entre les molécules. Le CRO peut être appliqué pour résoudre des problèmes dans les domaines discrets et continus. CRO a été exploré pour résoudre de nombreux problème d'optimisations, par exemple le travail de Throng et al [Throng et al., 2013] qui appliquer le CRO pour résoudre le problème du sac-à-dos binaire, [Sun et al., 2011] ont développé une approche hybride basé sur CRO et l'algorithme de recherche locale Lin-Kernighan pour résoudre le problème du voyageur de commerce mono-objectif. Par ailleurs, [Xu et al., 2013] ont développé une CRO basée sur la structure moléculaire double pour la planification du graphe acyclique orienté sur les systèmes informatique hétérogènes, récemment, [Saiffullah et Islam, 2016] ont employé les caractéristiques des réactions chimiques pour résoudre le problème de la plus courte super-séquence commune, qui est un problème NP-difficiles. A traves les différents travaux dans l'axe de CRO on constante que cet algorithme a montré une grande efficacité dans nombreux problèmes d'optimisations.
Dans ce qui va suivre, nous allons parler en détail sur les principes de cet algorithme et sa source d'inspiration.

1.7.1 Inspiration

L'algorithme inspiré des réactions chimiques est regis par les deux lois de la thermodynamique. La première loi est la conservation de l'énergie qui cite que l'énergie ne peut être ni crée ni détruite. L'énergie peut être transformée d'une forme à une autre et transféré d'une substance à une autre. Une réaction chimique est classée en deux types : une réaction endothermique ou exothermique. Une réaction endothermique est une réaction qui absorbe de la chaleur c'est-à-dire nécessite de la chaleur prévenant de l'environnement pour initialiser le processus de réaction, tandis qu'une réaction exothermique est une réaction qui dégage de la chaleur à l'environnement. La deuxième loi de la thermodynamique stipule que le degré de désordre dans un système augmente toujours. Cela signifie que l'énergie potentielle est stockée dans chaque molécule lorsqu'elle est convertie en une autre forme (énergie cinétique), le taux de désordre dans le système augmentera.

[Lam et Li, 2010] se sont inspirées de ce processus thermodynamique pour crée un algorithme bio-inspirés qui se base sur la collision entre les molécules. Chaque molécule M possède plusieurs attributs essentiels notamment sa structure moléculaire (ω) son energie potentielle (*PE*) et son énergie cinétique (*KE*). De plus, il existe de nombreuses variantes différentes de CRO en fonction des opérateurs de l'algorithme et de la manière dont ils sont utilisés pour le construire. La plupart des variantes CRO publiées incluent le nombre de collision que la molécule subite *NumHit*. La meilleure valeur que *M* a eu durant son existence *MinPE* et le nombre de collision associé à *MinPE* qui est *MinHitPE* qui représente la valeur de la fonction objectif de ω tandis que *KE* quantifie la tolérance du système d'accepter une mauvaise solution. [Sun et al., 2011].

Cet algorithme contient de types de collisions moléculaires : collision unimoléculaire qui est un processus dans lequel une seule molécule réagit afin de se transformer en une autre molécule, ou bien en plusieurs molécules. Cette collision donne deux types de réaction élémentaire qui sont la substitution simple et la décomposition, tandis qu'une collision intermoléculaire est lorsque plusieurs molécules entrent en collision les unes avec les autres. Cette collision aussi peut traduire deux types de collision élémentaire : la substitution double et la synthèse. Par conséquent, chaque collision peut produire quatre types de réactions élémentaires.

1.7.2 Les opérateurs chimiques

CRO supporte deux types de réactions chimiques : unimoléculaire et intermoléculaire. Les réactions unimoléculaires représentent une substitution simple et une décomposition par contre les réactions intermoléculaires entrainent une substitution double et une synthèse. Ces quatre opérateurs ont un rôle important dans l'exploration et l'exploitation de CRO. Puisque la synthèse et la décomposition ont un rôle d'exploration du domaine de recherche tandis que la substitution simple et double est l'exploitation des solutions trouvées. Les conditions responsables de l'acceptation ou de rejet des nouvelles solutions trouvées par ces opérateurs chimiques sont liées à la conservation de l'énergie.

- ***La substitution simple***

La substitution simple modélise la situation quand une molécule percute la paroi du conteneur et reste en une seule unité. Dans cette collision seulement la molécule existante ω était perturbée en ω' [Lam et Li, 2012]. Soit ω' une solution de l'entourage deω, et $PE\ (\omega')= f\ (\omega')$.L'énergie cinétique $KE\ \omega'$ est calculée selon la formule (1.1) :

$$KE_{\omega'} = (PE(\omega) - PE(\omega') + KE_{\omega}) \times \alpha. \quad (1.1)$$

De plus, l'énergie restante$(PE(\omega) - PE(\omega') + KE(\omega) \times (1 - \alpha))$, est transférée au buffer. Soit KELossRate un paramètre de CRO où : 0≤ KELossRate ≤1 et α est un nombre aléatoire dans [KELossRate, 1]. La molécule transformée sera accepté si la condition de conservation d'énergie de la substitution simple (Dans le cas de minimisation) est satisfaite comme exprimé dans la formule (1.2) :

$$PE(\omega) + KE_{\omega} \geq PE(\omega') \quad (1.2)$$

Cette condition accepte toujours les bonnes solutions$PE(\omega') > PE(\omega)$, mais dans le cas contraire $[PE(\omega) < PE(\omega')]$ elle peut accepter une mauvaise solution. Lorsqu'une molécule subit plus de réaction élémentaire elle aura plus de KE transféré dans le buffer. Par conséquence, la probabilité d'avoir une solution pire est plus faible lors d'un changement ultérieur.

- ***La décomposition***

La décomposition représente la situation où une molécule percute la paroi du conteneur et ensuite se décompose en plusieurs parties (pour simplifier, nous considérons deux parties). Supposons que $M\omega$ produire $M\omega 1'$ et $M\omega 2'$.

La décomposition est un moyen pour le système d'explorer d'autres régions de l'espace des solutions après une recherche locale suffisante par les collisions inefficaces, théoriquement, même la génération de solutions indépendantes de celle existante (génération aléatoire d'une nouvelle solution) est faisable. Pour accepter la nouvelle molécule, la formule 1.3 doit être satisfaite.

$$PE(\omega) + KE_{\omega} + \delta_1\delta_2 \text{ buffer } \geq PE(\omega_1') + PE(\omega_2') \tag{1.3}$$

Où δ_1 et δ_2 sont deux nombres indépendants appartient à l'intervalle [0,1]. Pour supporter le changement moléculaire requis pour la réaction une partie de l'énergie du buffer est nécessaire comme suit :

$$\text{buffer'} = (1 - \delta_1\delta_2) \text{ buffer} \tag{1.4}$$

Si la formule (1.3) est vérifiée alors $M\omega$ est remplacé par les deux nouvelles molécules générées et leurs énergies cinétiques partagent aléatoirement l'énergie restante. [Lam et Li, 2010].

$$E_{\text{dec}} = (PE_{\omega} + KE_{\omega} \times \delta_1 \times \delta_2 \times \text{ buffer }) - \left(PE_{\omega_1'} - PE_{\omega_2'}\right)$$

$$\text{Et} \begin{cases} KE_{\omega_1'} = E_{dec} \times \delta_3 \\ KE_{\omega_2'} = E_{dec} \times (1 - \delta_3) \end{cases} \tag{1.5}$$

Où δ_3 est un nombre aléatoire généré dans [0,1].

- ***La substitution double***

La substitution double se produit lorsque plusieurs molécules (généralement deux $M\omega 1$ et $M\omega 2$) entrent en collision les unes les autres, puis rebondissent. Cette réaction élémentaire est très similaire à la substitution simple, car elle perturbe les molécules existantes $\omega 1$ et $\omega 2$ en $\omega 1'$ et $\omega 2'$, mais aucune énergie de buffer (E_{inter}) utilisé. La condition de conservation de l'énergie est exprimée comme suit :

$$PE_{\omega_1} + KE_{\omega_1} + PE_{\omega_2} + KE_{\omega_2} \geq PE_{\omega_1'} + PE_{\omega_2} \tag{1.6}$$

Pour que les changements apportés par cette réaction chimique seront acceptés. La condition 1.6 doit être satisfaite [Lam et Li, 2010]. Les énergies cinétiques des molécules transformées partagent l'énergie restante comme suit :

$$E_{\text{inter}} = \left(PE_{\omega_1} + KE_{\omega_1} + PE_{\omega_2} + KE_{\omega_2}\right) - \left(PE_{\omega_1'} + PE_{\omega_2}{}'\right)$$

$$\text{Et} \begin{cases} KE_{\omega_1'} = E_{\text{inter}} \times \delta_4 \\ KE_{\omega_2'} = E_{\text{inter}} \times (1 - \delta_4) \end{cases} \tag{1.7}$$

Où δ_4 est un nombre aléatoire généré dans [0,1] et E_{inter} est l'énergie de buffer.

❖ ***La synthèse***

Cet opérateur représente plus d'une molécule (généralement deux molécules $M\omega 1$ et $M\omega 2$) qui entrent en collision et se combinent en donnent une nouvelle molécule $M\omega'$. L'idée de la synthèse est la diversification des solutions. La condition de conservation d'énergie de la synthèse [Lam et Li, 2010] est indiquée dans la formule 1.8 :

$$PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'} \tag{1.8}$$

Si la condition (1.8) est vérifiée, le résultat $KE\omega'$ prend tout le reste de l'énergie.

$$KE_{\omega'} = \left(PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}\right) - (PE_{\omega'}) \tag{1.9}$$

1.7.3 Schéma général de l'algorithme

Tout comme les autres algorithmes évolutionnaires ou méta-heuristique, CRO est flexible et peut être adapté à une variété de problèmes.

CRO est un processus en trois étapes principales : L'initialisation, les itérations et l'étape finale. Lors de l'initialisation CRO définit les molécules et les paramètres nécessaires à son fonctionnement, et explore l'espace des solutions au cours des itérations. L'algorithme atteint finalement l'étape finale où il produira sa meilleure solution. Le schéma général de processus de CRO est présenté dans la figure (1.5)

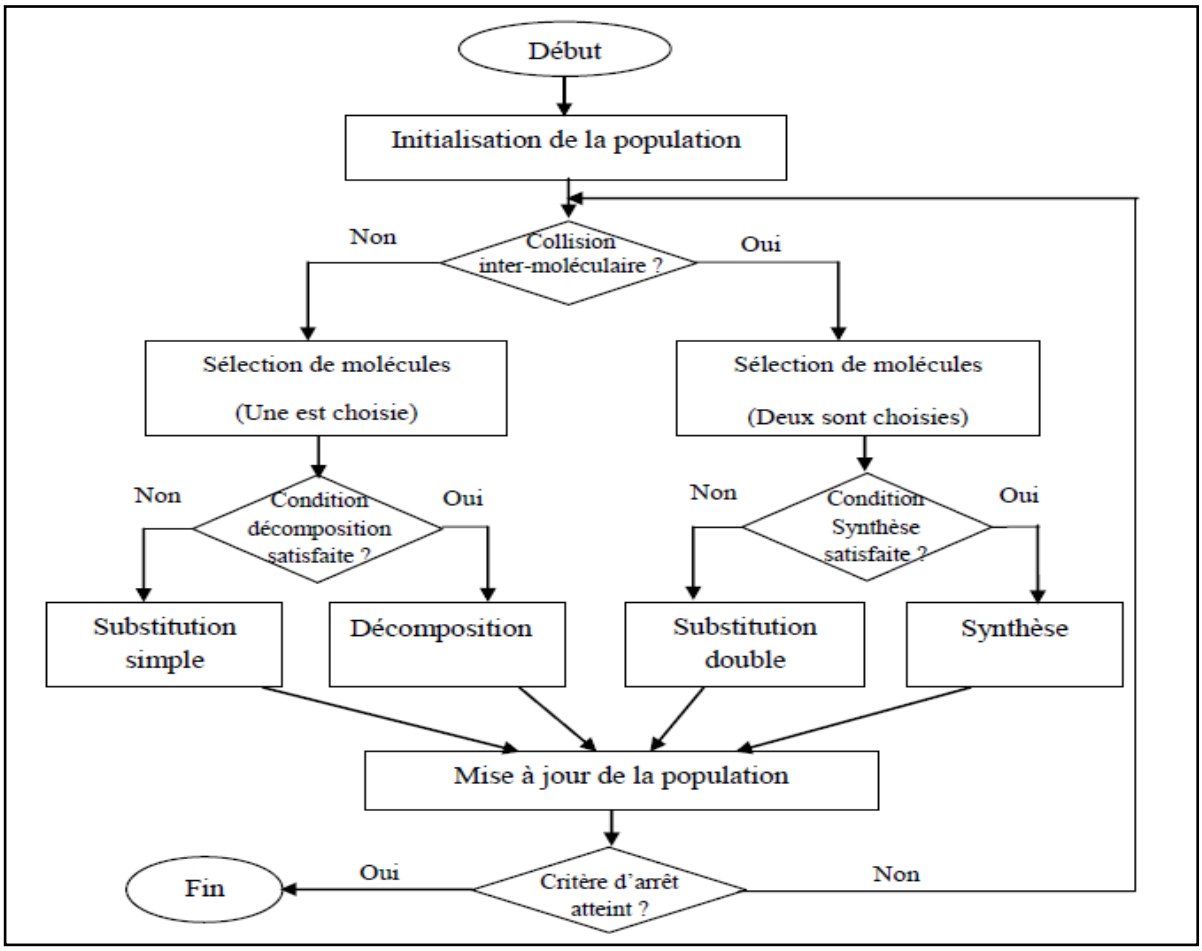

Figure 1.5 : Schéma général de CRO [Lam et Li, 2010]

- **L'initialisation**

CRO commence par générer une population de molécules PopSize. Chaque molécule est composée des champs mentionnés précédemment. De plus, cet algorithme initialise les paramètres algorithmiques qui sont KELossRate, MoleColl, buffer, InitialKE, α et β.

- **Les itérations**

Dans chaque itération de CRO, les molécules avec leur énergie se déplacent et déclenchent des collisions, une molécule peut soit percuté la paroi du conteneur soit entrée en collision les unes avec les autres. Ceci est déterminé en générant un nombre aléatoire r dans [0 ,1]. Si r > MoleColl ou si le système n'a qu'une seule molécule, on a une collision unimoléculaire, sinon une collision intermoléculaire se produiront.

Pour que la collision unimoléculaire se produise, la molécule M doit pouvoir subir une décomposition ou une substitution simple. Si la condition de décomposition est satisfaite (NumHit-MinHit > α), alors M sera remplacé par M_0 et M_1, si la condition de conservation de l'énergie de décomposition est satisfaite (formule 1.3). Sinon, nous obtenus une substitution simple et le changement de la molécule M sera accepté sauf si la condition (1.2) est vérifiée.

De même, pour une collision intermoléculaire, on sélectionne au hasard deux molécules M_1 et M_2 ces molécules subiront une synthèse ou une substitution double. Dans le cas où la condition de la synthèse (KE$\leq \beta$) est vérifiée alors une synthèse sera appliqué et les molécules M_1 et M_2 seront remplacées automatiquement par la nouvelle molécule M' si la condition de conservation de l'énergie (1.8) est satisfaite. Sinon, la substitution double aura lieu, le changement engendré par l'opérateur sur les deux molécules seront acceptés si la condition de conservation d'énergie (1.6) est satisfaite.

Après la fin d'une réaction élémentaire. L'algorithme vérifié si la condition de conservation de l'énergie est respectée. Dans le cas contraire le changement est annulé. Ensuite, l'algorithme vérifie si la nouvelle solution a une valeur de fonction objectif meilleure que l'ancienne. Si c'est le cas, alors elle sera enregistrée, si aucun critère d'arrêt n'est atteint une nouvelle itération sera lancée.

- **L'étape finale**

Si un critère d'arrêt est satisfait, l'algorithme passe à l'étape finale. Les critères d'arrêt sont définis en fonction des besoins et des préférences de l'utilisateur. Elles incluent la quantité maximale de temps CPU utilisé, l'obtention d'une valeur de fonction objectif inférieure au seuil prédéfini, le nombre maximal d'itérations effectués sans amélioration…etc. A ce stade, le processus de CRO se terminera par l'affichage de la meilleure solution trouvé avec sa valeur de fonction objectif.

1.8 Conclusion

Dans ce chapitre, nous avons introduit le contexte de notre travail. Nous avons tout d'abord présenté quelques notions de base de l'optimisation, ensuite nous avons introduit le paradigme de la complexité. En outre, nous avons essayé de présenter un état de l'art sur les méthodes de résolution de différents problèmes d'optimisation présentées dans la littérature. En commençant par les méthodes exactes, en citant quelques méthodes, leurs avantages et inconvénients. Ensuite nous avons passé vers les méthodes approchées, nous avons donné une description

générale des approches heuristiques et méta-heuristiques, nous avons présenté leurs concepts communs, leurs avantages et inconvénients et nous avons cité quelques exemples de ces méthodes.

Dans le chapitre qui suit, nous présentons le problème de tournées de véhicules multi-dépôts (MDVRP), et les principales méthodes proposées pour le résoudre.

Chapitre 2

Le Problème de Tournées de Véhicules Multi-dépôts

2.1 Introduction

Dans plusieurs sphères de la société, on recherche la maximisation de l'efficacité avec un minimum d'investissement. En raison des nombreuses options offertes aux clients, ils deviennent de plus en plus exigeants, satisfaire les clients tout en réduisant les coûts est une tâche difficile à réaliser.

L'étude systématique de ces problématiques par la modélisation mathématique et l'élaboration de méthodes de résolution facilitent l'atteinte de ce but. La recherche opérationnelle tente, dans cette optique, d'analyser et de formuler ces problématiques, permettant ainsi une étude approfondie des principaux facteurs qui sont à la base de celles-ci.

La logistique du transport représente l'une des branches de la recherche opérationnelle ayant reçu le plus d'attention de la part des chercheurs. Les problèmes de transport sont l'un des problèmes importants pour de nombreux domaines tels que l'économie, l'industrie, la production... etc. qui doivent trouver des solutions optimales. Notamment, dans les problèmes de tournées de véhicules multi-dépôts (MDVRP) qui est un cas particulier de problème de tournées de véhicules (VRP) classique. Dans le MDVRP les véhicules quittent les dépôts pour servir certains clients selon certaines contraintes pour les satisfaire et avec un coût minimum.

Dans ce chapitre, nous présentons la définition de problème de tournées de véhicules (VRP) classique, ses variantes. Ensuite, nous nous focalisons sur le problème de tournées de véhicules multi-dépôts (MDVRP), sa définition mathématique, les différentes méthodes de résolution développées dans la littérature pour le résoudre.

2.2 Le problème de tournées de véhicules

Le problème de tournées de véhicules (VRP) (aussi appelé VRP pour Vehicle Routing Problem) est un problème d'optimisation combinatoire. Il s'agit de déterminer les tournées d'une flotte de véhicules afin de livrer une liste de clients, ou de réaliser des tournées d'interventions (maintenance, réparation, contrôles) ou de visites (visites médicales, commerciales, etc.). Le but est de minimiser le coût de livraison des biens. Ce problème est une extension classique du problème du voyageur de commerce [Dhaenens et al., 2002] lorsque la contrainte de capacité sur les véhicules doit être explicitement prise en compte. Le problème de VRP fait partie de la classe des problèmes NP-complet, il a constitué un défi pour un bon nombre de chercheurs, comme détaillé par Laporte [Laporte, 2009] chez son rapport sur plus de 50 ans de travail dans le domaine. L'état de l'art permet actuellement de résoudre des problèmes dont la taille atteint 200 clients et 17 véhicules, comme cela est décrit par [Baldacci et al.,2011]. La figure 2.1 donne un exemple de ce problème.

On peut définir le problème de tournées de véhicules (VRP) comme : une flotte de véhicules localisées à un dépôt (centre de distribution) les besoins d'un ensemble de clients doivent être satisfaits, tant au niveau de la livraison que de la collecte des marchandises. (Dans la version classique de VRP, le problème est résolu soit dans le cadre de la livraison, soit dans le cadre de la collecte mais pas les deux). Tous les clients possèdent une demande et celle-ci doit être satisfaite par un et un seul véhicule. La flotte peut être homogène ou non (homogène : les éléments Composés de même nature, semblables). Une tournée au plus est assignée à chaque

véhicule, cette tournée représente l'ordonnancement des clients à visiter au cours de la période de planification.

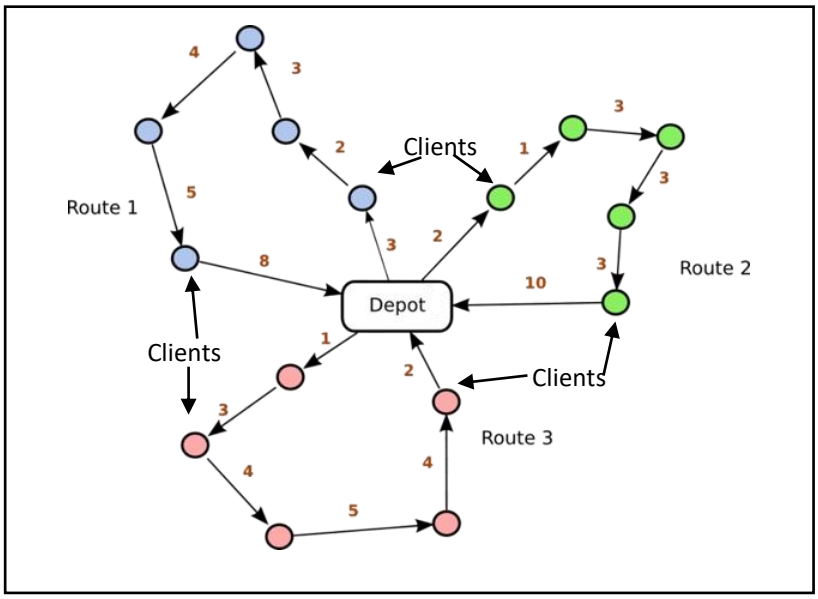

Figure 2.1. Exemple illustratif du problème de tournées de véhicules [Web4].

De plus, la somme des demandes des clients d'une tournée donnée ne doit pas dépasser la capacité du véhicule. Finalement, chaque tournée doit débuter et se terminer au même dépôt. Il faut donc être en mesure d'affecter chaque client à la tournée de l'un des véhicules et de déterminer la meilleure façon d'effectuer chacune d'entre elles. Autrement dit, l'ordre dans lequel les clients seront visités, afin de minimiser le coût total de transport (minimiser la distance totale parcourue). Plusieurs contraintes peuvent être introduites dans le problème, comme l'ajout d'une durée de livraison à chaque client et d'une durée maximale sur chacune des tournées, ou de fenêtres de temps (périodes précises) pour livrer les clients.

Le problème peut être modélisé sous forme de graphe complet G = (V, E) où V = {v_0, v_1, …, v} est un ensemble de sommets modélisant les villes (ou les clients) et E = {(v_i ; v_j) : v_i ; v_j ϵ V, $i < j$} est un ensemble d'arêtes ou d'arcs reliant les sommets. Un autre sommet v0, correspondant au dépôt.

Nous supposons que le graphe G = (V ; E) est complet (tous les sommets sont reliés entre eux), n nombre de clients (ou sommets), m nombre de véhicules, Q représente la capacité des véhicules, qi est la demande du client i, c_{ij} le coût de l'arête entre les sommets i et j (distance ou temps de parcours). Cette formulation traduit la modélisation naturelle du problème par la définition une variable binaire *xijk* égale à 1 si le véhicule *k* parcourt l'arc (v_i, v_j), *xijk* définit comme suit :

$$x_{ijk} = \begin{cases} 1 \text{ si } (i,j) \text{ est parcouru par le véhicule } k \text{ ;} \\ 0 \text{ sinon.} \end{cases}$$

Enfin, le problème de tournées de véhicules (VRP) peut être résumé comme suit :

- Un distributeur doit envoyer des véhicules avec les demandes aux clients.
- La demande de chaque client doit être satisfaite par un véhicule.
- Déterminez la route que chaque véhicule devra suivre.
- La tournée servie ne dépasse pas la capacité de véhicule.
- Afin de minimiser le coût total de livraison, il est nécessaire de minimisée la distance totale.

2.2.1 Les variantes du VRP

La variation des paramètres du VRP, la suppression et/ou l'ajout ou bien la combinaison de contraintes du VRP classique permettent de définir un ensemble de variantes du VRP. L'apparition de nouvelles variantes du VRP est due essentiellement aux activités des chercheurs qui tentent de mettre à profit les ressources puissantes disponibles pour être de plus en plus fidèles aux problèmes rencontrés dans l'industriel. En particulier, ils s'intéressent non seulement aux problèmes sur nœuds (dans lesquels les clients à desservir sont sur les nœuds) mais également aux problèmes sur arcs. Dans ce qui suit, nous présentons une répartition des variantes du VRP selon le type de contraintes :

- **Problème de tournées de véhicules capacitaire (CVRP) :**
  Le problème de tournées de véhicules capacitaire (CVRP) est l'une des variantes les plus importantes dans la classe des problèmes VRP. CVRP a été proposé pour la première fois par Dantzig and Ramser [Dantzig et Ramser, 1959] sous le nom de "track dispatching problem". Il consiste à affecter chaque client (chacun ayant une demande connue) à une tournée effectuée par un seul véhicule de capacité finie (réalisant au moindre coût le trajet total). Ce véhicule commence et se termine sa tournée au dépôt [Ralphs, 2003]. Les solutions trouvées doivent respecter des contraintes de capacité fixe pour chaque véhicule.

- **Problème de tournées de véhicules avec un chargement complet (VRPFL) :**
  C'est un VRP avec utilisation complète de la capacité du véhicule.

- **Problème de tournées de véhicules avec flotte hétérogène (VRPHF) :**
  Pour ce problème la flotte est composée de véhicules de types différents, qui se distinguent par la capacité, la puissance, le coût de transport . . . etc.

- **Problème de tournées de véhicules avec backhauls (VRPB) :**
  Ce problème comporte deux types de clients, soit receveurs (linehauls), soit livreurs (backhauls). Tous les produits livrés sont pris d'un dépôt et tous les produits prélevés sont retournés au dépôt. Ce modèle général résulte en trois catégories de VRPB, Parragh et al [Parragh et al., 2006] :

  -Le VRP with Clustered Backhauls (VRPCB):
  Dans lequel toutes les livraisons sont effectuées avant le premier ramassage, comme dans le travail de Brandao [Brandao, 2006].

- Le VRP with Mixed Linehauls and Backhauls (VRPMB):
  Dans lequel les ramassages et les livraisons peuvent être mêlés dans une même tournée, comme dans le travail de Crispin et Brandao [Crispin et Brandao, 2005].
-Le VRP with Simultaneous Delivery and Pickup (VRPSDP):
  Dans lequel chaque client peut être receveur et livreur en même temps, Hoff et Lokketangen en 2006 [Hoff et Lokketangen. 2006]

❖ **Problème de tournées de véhicule ouvert (OVRP) :**
Ce problème est identique au VRP, seulement le véhicule est libre de rejoindre ou pas le dépôt après la fin de la tournée. S'il choisit de rejoindre le dépôt, il doit reprendre le parcours de la tournée dans le sens inverse.

❖ **Le problème de tournées de véhicules avec fenêtres temps (VRPTW) :**
Le problème d'acheminement de véhicules avec fenêtres horaires (VRPTW) peut être défini comme le choix d'itinéraires pour un nombre limité de véhicules afin de desservir un groupe de clients dans les fenêtres de temps (la planification de visites chez des clients qui sont disponibles que pendant des périodes spécifiques). Chaque véhicule a une capacité limitée. Il part du dépôt et se termine au dépôt. Chaque client doit être servi exactement une fois.

❖ **VRP à Contraintes Liées à la Demande des Clients :**
Dans cette variante, il existe deux types de demande des clients :
-**VRP à Demande Déterministe** C'est un problème fréquent pour les entreprises qui font des livraisons sur commande. En effet, le livreur connait avant son départ du dépôt la quantité à livrer à chacun de ses clients.
-VRP à Demande Stochastique Contrairement au précédent, dans le VRP avec demande stochastique, le livreur ne connait pas la quantité à livrer au client, il va la découvrir au moment de le servir. Il estime approximativement la demande de chaque client par une fonction stochastique.

❖ **Problème de tournées de Véhicules Multi-Périodique (MPVRP) :**
Problème de tournées de Véhicules Multi-Périodique (MPVRP) considère un horizon de planification à M périodes. Chaque client doit être visité k fois au cours de l'horizon ($1 \leq k \leq M$). Et les demandes quotidiennes sont fixes [Lacomme et al. 2005], Francis et Smilowitz [Francis et Smilowitz, 2006].

❖ **Problème de tournées de Véhicule à Produits Multiples (MPVRP) :**
Une gamme de produits doit être livrée aux différents clients par chaque véhicule en une seule tournée.

❖ **Problème de tournées de véhicules sélectives (TOP) :**
Le problème de tournées de véhicules sélectives (TOP "Team Orenteering Problem") a d'abord été étudié par [Chao et al., 1996] et il appartient à la famille des problèmes de tournées de véhicules avec profit. Dans cette catégorie de problèmes, il est à priori

impossible de visiter tous les clients en raison de ressources limitées. On associe plutôt un profit à chaque client qui représente sa valeur. Ce profit est collecté lorsque le client est visité par l'un des véhicules disponibles. L'objectif est de sélectionner un sous ensemble de clients à servir tout en maximisant le profit total collecté.

- **Problème de tournées de véhicules dynamique (DVRP) :**
  Dans le VRP classique, toutes les demandes pour tous les clients sont connues à priori. Cependant, de nombreux problèmes de notre vie pratique incluent au moins un (ou plusieurs) caractère dynamique comme par exemple l'apparition d'un nouveau client en cours de la journée. Dans ce cas, le décideur doit changer la planification des véhicules en réponse aux nouvelles demandes qui arrivent au cours du temps [Kilby et al., 1998].

- **Le problème de tournées de véhicules multi-dépôts (MDVRP) :**
  Comporte plusieurs dépôts dans lesquels sont localisés les véhicules. Chaque tournée d'un véhicule doit commencer et se terminer au même dépôt. Chaque client doit être visité exactement une fois par l'un des véhicules situés dans les dépôts qui contient les demandes de ce client [Mingozzi, 2005].

Il existe de nombreux articles dans la littérature qui étudient des problèmes combinant plusieurs variantes du problème de tournées de véhicules.

Dans notre mémoire, nous nous sommes intéressés d'étudier le problème de tournées de véhicules multi-dépôts.

2.3 Le problème de tournées de véhicules multi-dépôts

Le problème de tournées de véhicules multi-dépôts (**MDVRP**) est un problème célèbre formulé en 1959 par Dartzig et Ramser [Dantzig et Ramser, 1959]. MDVRP est considéré comme un problème NP-difficiles. De plus, ce problème est important et difficile dans la gestion de la logistique.

MDVRP est un cas particulier de problème de tournées de véhicules (VRP) classique. Ce problème impose l'affectation des clients aux dépôts et l'acheminement des véhicules pour visiter ces clients une seule fois, par un seul véhicule partant d'un seul dépôt. Chaque véhicule part d'un dépôt, sert les clients affectées à ce dépôt et fait le retour au même dépôt de départ.

L'objectif du MDVRP est de minimiser la distance totale de la livraison de tous les clients. En outre, l'objectif peut également être la minimisation du nombre de véhicules nécessaires, moins de véhicules signifient que le coût total est réduit. Il existe de nombreuses applications en temps réel qui ont motivé la recherche dans le domaine du MDVRP, telles que les urgences, la distribution de journaux, les services de taxi et la gestion de la collecte des ordures. La figure 2.2 montre un exemple illustratif du problème MDVRP :

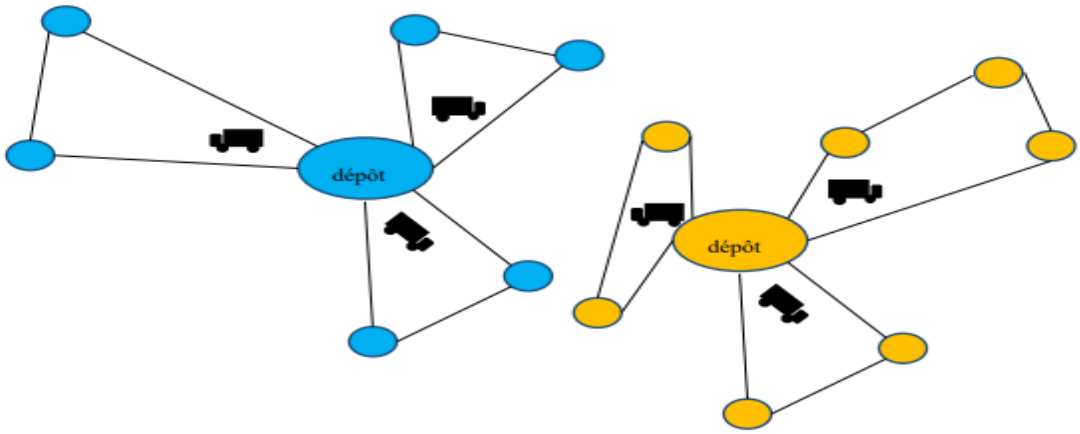

Figure 2.2 : Exemple illustratif du MDVRP

Le MDVRP est beaucoup plus avantageux que le VRP, il est plus réaliste vu le nombre de dépôts ouverts pour servir les clients.

2.4 Formulation mathématique du MDVRP

L'objectif de ce problème est de concevoir l'ensemble de tournées affectées pour chaque dépôt afin de servir les différents clients, les demandes des clients sont déterministes et connues.

On cherche donc à insérer chaque client sur une tournées de la relation d'un et d'un seul véhicule de façon à minimiser la distance totale parcourue, tout en respectant le fait que chaque tournée doit débuter et se terminer au même dépôt. De plus, chacune tournées qui composent ces relations doit respecter les contraintes de capacités. Les notations et formulation mathématique sont les suivantes :

Tableau 1.1: Formulation mathématique du MDVRP. [Mirabi et al., 2010].

	Modèle Mathématique	Description
Ensembles	I	Ensemble de tous les dépôts
	J	Ensemble de tous les clients
	K	Ensemble de tous les véhicules
Paramètres	N	Nombre des clients
	C_{ij}	Distance entre le point i et j, i,j$\in I \cup J$
	V_i	Débit maximal au dépôt i
	D_j	Demande du client j
	Q_k	Capacité du véhicule sir l'itinéraire k
Variable de décision	X_{ijk}	1, si le point i possède immédiatement le point j sur la route k (i,j$\in I \cup J$); 0, sinon.
	Zij	1, si le client j est affecté au dépôt i ; 0, sinon.
	U_{lk}	Variable auxiliaire pour les contraintes d'élimination des sous-tours dans l'itinéraire.
Contraintes	$Min\ z = \sum_{i \in J \cup j} \sum_{j \in I \cup J} \sum_{k \in K} C_{ij} x_{ijk}$	La fonction objectif minimise la distance totale de la livraison
	$\sum_{k \in K} \sum_{i \in I \cup J} x_{ijk} = 1, j \in J$	exige que chaque client soit affecté à une seule route
	$\sum_{j \in J} d_j \sum_{i \in I \cup J} x_{ijk} \leq Q_k, k \in K$	est la contrainte de capacité fixée pour les véhicules
	$U_{IK} - U_{jk} + N x_{Ijk} \leq N - 1, l, j \in J, k \in K$	est le nouveau ensemble de contraintes d'élimination de sous-tours (tournées incomplet).
	$\sum_{j \in I \cup j} x_{ijk} - \sum_{j \in I \cup j} x_{jik} = 0, k \in K, i \in I \cup J$	La contrainte de conservation de débit
	$\sum_{i \in I} \sum_{j \in J} x_{ijk} \leq 1, k \in K$	Assurer que chaque itinéraire peut être servi au plus une fois
	$\sum_{j \in J} d_i z_{ij} \leq V_i, i \in I$	Les contraintes de capacité pour les dépôts
	$-z_{ij} + \sum_{u \in I \cup J} (x_{iuk} + x_{ujk}) \leq 1, i \in I, j \in J, k \in K$	précise qu'un client peut être affecté à un dépôt uniquement s'il existe un itinéraire à partir de ce dépôt passant par ce client
	$x_{ijk} \in \{0,1\}, i \in I, j \in J, k \in K$	les exigences binaires sur les variables de décision
	$z_{ij} \in \{0,1\}, i \in I, j \in J$	les exigences binaires sur les variables de décision
	$U_{Ik} \geq 0, l \in J, k \in K$	Les Ulk les variables auxiliaires prenant des valeurs positives

2.5 Les méthodes de résolution de MDVRP

Le MDVRP est l'un des problèmes le plus étudié dans le domaine de la recherche opérationnelle, le problème a attiré l'attention de beaucoup de chercheures dans la littérature. Au cours des dernières années avec l'évolution des méthodes d'optimisation développés dans le domaine de la recherche opérationnelle, plusieurs études sur le MDVRP ont opté sur le développement de méthodes de résolutions de ce problème. Ces méthodes de résolution sont généralement réparties en deux parties : les méthodes exactes et les méthodes approchées. Ces dernières sont divisées en deux sous classes qui sont les heuristiques et les méta-heuristiques.

Plusieurs méthodes exactes ont été proposées par différents chercheurs, par contre, d'autres travaux ont mentionnée que pour l'instant les meilleures méthodes exactes pour MDVRP sont toujours cantonnées à des problèmes de taille relativement restreinte. En effet, plusieurs travaux dans la littérature ont basé sur les méthodes approchées pour résoudre ce problème. Dans ce qui suit, nous allons citer quelques exemples de chaque classe de ces méthodes.

2.5.1 Les méthodes exactes

Cette classe contient des algorithmes pour résoudre MDVRP de manière exacte.

Il est possible de répertorier dans la littérature quelques méthodes exactes permettant de résoudre ce problème. Le premier papier qui visait la résolution exacte du MDVRP a été introduit en 1984 par Laporte, Nobert et Arpin [Laporte, Nobert et Arpin, 1984].

[Laporte et al., 1984] ont proposé un algorithme exacte basé sur une méthode de séparation et d'évaluation progressive (branch-and-bound) afin d'effectuer la résolution du MDVRP. En relaxant certaines contraintes du modèle général (sous-tours, intégralité, contraintes de chaîne) et en générant, tout au long de l'algorithme, des coupes lorsque certaines de ces cotraitantes sont violées. Les auteurs ont pu résoudre de manière optimale des problèmes créés au hasard avec plus ou moins 40 clients. Les contraintes de chaîne (la tournée) sont construites de façon à ce que deux dépôts ne puissent être reliés par une chaîne de sommets et s'assurent que chaque tournée débute et se termine au même dépôt. Un autre algorithme utilisant une approche par séparation et évaluation progressive (SEP) est développé par Laporte, Nobert et Taillefer [Laporte., 1988]. Dans ce travail, il obtient un problème d'affectation avec certaines contraintes à partir du problème initial en transformant le graphe. Par l'implantation d'une version modifiée de l'algorithme de SEP de Carpaneto et Toth [Carpaneto et Toth, 1980], une série de problèmes d'affectation avec contraintes sont résolus. Ces derniers sont générés à partir d'une solution relaxée du problème initial ainsi qu'à chaque nœud de l'arbre afin de respecter les différentes Contraintes. Plusieurs instances allant jusqu'à 80 clients et 3 dépôts ont été résolues à l'optimum.

[Toth et al. 2002] ont proposé une borne inférieure pour le MDVRP à l'aide une méthode de Programmation Linéaire en nombres entiers (PLNE) lorsque le nombre de contraintes est trop élevé, les méthodes précédentes ne sont plus applicables. Les méthodes de type Branch & Cut n'incluent pas de contraintes sur la transformation pas à pas du problème (c'est-à-dire les phases de coupe dans l'espace des solutions), dont les contraintes d'intégrité ont été assouplies sur les variables entières. Un solveur PL (Programmation Linéaire) est utilisé pour tenter de trouver une solution optimale entière qui respecte les contraintes du problème PLNE (Programmation

Linéaire en Nombres Entiers). Dans le cas contraire, une phase de décomposition (i.e. Branch) du problème en 2 sous problèmes est nécessaire et la phase de coupes est relancée sur ces sous problèmes.

On trouve aussi une méthode exacte développé par [Baldacci et Mingozzi ,2009] pour résoudre le problème de tournées de véhicules hétérogènes (HVRP) qui est capable de résoudre, entre autres problèmes, le MDVRP. Cet algorithme est basé sur la formulation de partitionnement d'ensembles, où une procédure est d'abord appliquée pour générer des routes, suivie de trois procédures de délimitation pour réduire le nombre de variables. Dans la formulation de partitionnement d'ensemble, les variables représentent les routes précédemment générées. Les auteurs ont présenté les résultats de calcul pour les instances MDVRP avec jusqu'à 200 clients et 2 à 5 dépôts.

D'autre part, [Contardo et Martinelli, 2014] ont donné une présentation d'une nouvelle méthode exacte pour le MDVRP basée sur la fixation de variables, la génération de colonnes et de coupes et l'énumération de colonnes. La méthode proposée a pu prouver l'optimalité pour la première fois pour certaines instances de benchmark utilisé.

Pour le MDVRP, il existe encore peu d'algorithmes exacts disponibles dans la littérature, alors que plusieurs procédures heuristiques ont déjà été proposées. Selon la revue sur le MDVRP proposée par Montoya-Torres et al [Montoya-Torres et al,2015], 25 % des travaux sur MDVRP utilisaient des techniques de résolution exactes, tandis que les 75 % restants des travaux utilisaient des techniques heuristiques ou méta-heuristiques (pour des instances de plus grande taille) qui ne visent pas à trouver la solution optimale exacte mais une approximation proche de l'optimale en temps raisonnable.

2.5.2 Les méthodes approchées

Vu les limites des méthodes exactes de trouver des solutions optimales dans un délai raisonnable pour les grandes instances de MDVRP, les méthodes approchées (heuristiques et métaheuristiques) ont tiré l'attention des chercheurs. Dans la littérature, plusieurs méthodes ont été développées pour ce problème. On donne par la suite quelques travaux basés sur les méthodes approchées.

2.5.2.1 les heuristiques

Les méthodes heuristiques nous permettent de trouver des solutions « approchées » pour des problèmes d'optimisation. Ces solutions ne sont pas forcément optimales mais peuvent être obtenus dans un temps relativement correct. Dans cette section nous présentons les travaux existants liés aux solutions MDVRP par divers méthodes heuristiques.

Une des premières références aux problèmes de tournées de véhicules multi-dépôts est due à]Tellman, 1969]. Cet auteur reprend l'idée formulé par Clarke and Wright, après avoir fait une affectation initiale de chaque client à son dépôt le plus proche, une tournée aller-retour chez chacun d'entre eux est construite. Puis, la fusion de tournée selon le critère du plus grand gain possible. Dans le même esprit, [Tellman et Hiering, 1971] ont proposé en 1971 un algorithme qui basé sur l'évaluation des possibilités de fusion avant de faire un choix, les auteurs ont rapporté cependant que cette procédure exige un temps de calcul important. [Tellman et Caïn,

1972] ont repris les formulations développées par Tellman en 1971 et ont élaboré un algorithme basé sur le principe d'évaluation des fusions dans un arbre d'énumération ou à chaque itération une borne supérieur sur les gains possibles de fusions de tournées est évaluée.

Toujours dans la même période [Wren et Holiday, 1972], ont proposé un algorithme qui se base sur une construction d'une solution initiale par une méthode de balayage. Le principe de cette technique est simple : on balaye l'espace des clients avec un axe dont l'extrémité se trouve au dépôt et on ajoute les clients aux itinéraires des véhicules dans l'ordre où ils sont rencontrés et tant qu'ils ne font pas accéder la capacité maximale du véhicules. Les auteurs dans ce travail [Gillett et Johnson, 1976] ont utilisé une procédure de clustering, c'est une heuristique de balayage pour chaque dépôt, l'idée est de regrouper les clients de façon à former une tournée autour de chaque dépôt.

[Chao et al., 1993] ont développé une nouvelle approche pour résoudre MDVRP, basée sur la méthode des registres (record-to-record) de [Dueck, 1993], l'algorithme met l'emphase sur le raffinement d'une solution initiale. Cette dernier étant généré rapidement en assignat chaque client à son dépôt le plus proche. Les auteurs ont démontré les performances de cette méthode sur les problèmes publié par [Gillet et Johnson, 1976], ainsi que sur de nouvelles instances allant jusqu'à neuf dépôts et 360 clients. L'heuristique de Chao, Golden et Wasil surpasse les algorithmes de [Gillet et Johnson, 1976], [Golden, Mangnanti et Nguyan., 1977] ainsi que [Raft, 1982] sur l'ensemble des 11 problèmes.

Par ailleurs, [Raft, 1982] a proposé une approche multi-phases avec un raffinement supplémentaire. L'idée de cette approche est tout d'abord, d'évaluer le nombre de tournées nécessaires afin d'effectuer l'ensemble de la distribution, en divisant la demande totale par la capacité d'un véhicule. Dans une deuxième étape l'heuristique estime un centre pour chaque tournée. Par la suite, chaque client se voit affecté selon un critère de pénalité, à l'un de ces centres. Une itération s'effectué entre les deux dernières jusqu'à ce qu'une stabilité s'établisse au niveau de la position des centres. Chaque combinaison centre-clients est ensuite assignée au dépôt le plus proche par rapport au centre. L'auteur utilise l'algorithme pour le PVC à m-tours de [Russel, 1977] afin de déterminer les tournées à chacun des dépôts. Finalement, un raffinement de la solution est effectué à l'aide de la méthode 3-opt de [Lin, 1965].

Plusieurs autres modules sont discutés par Raft. Par exemple, il a considéré le cas ou des contraintes sur les périodes de livraison chez différentes clients sont présentées. Raft expérimente et comparé son algorithme avec quelques instances proposées par Gillett et Johnson en 1976 [Gillett et Johnson, 1976] ainsi que [Christofides et Eilon, 1969]. Les plus larges instances comportent cinq dépôts et 249 clients sur quatre problèmes, Raft obtient deux fois la meilleure solution. Cependant, l'une des deux ne respecte pas la contrainte de longueur des routes.

[Salhi et Sari, 1997] ont proposé une méthode heuristique à trois niveaux pour résoudre le MDVRP. Le premier niveau fait la construction d'une première solution réalisable. Le deuxième et troisième niveau fait l'amélioration des tournées de chaque dépôt (intra-dépôt) et l'amélioration des tournées de tous les dépôts (inter-dépôt).

Une variante du MDVRP, le min-max Split Delivery MDVRP with Minimum Service Time Requirement (min-max SDMDVRP-MSTR) a été résolu en utilisant une approche heuristique par [Wang et al., 2016]. Dans lequel opèrent en trois étapes :

1) Initialiser une solution réalisable sans scission

2) Améliorer les itinéraires les plus longs en fractionnant les temps de service

3) Assurer que toutes les exigences de temps de service minimum sont satisfaites

2.5.2.2 Les méta-heuristiques

Dans cette section, nous citons quelques solutions méta-heuristiques qui ont été proposé pour résoudre le MDVRP.

En 1996, Renaud, Laporte et Boctor [Renaud et al., 1996] ont proposé une nouvelle méta-heuristique pour résoudre MDVRP en utilisant la recherche tabou avec restriction d'itinéraire et de capacité.

[Beatrice et al., 2009] ont proposé une application de l'approche des algorithmes génétiques pour le MDVRP. Le GA proposé utilise un codage indirect et une stratégie adaptative d'échange de mutations inter-dépôts pour le MDVRP avec des restrictions de capacité et de longueur de tournées. Les auteurs ont montré la performance de cet algorithme sur un ensemble de 23 problèmes de référence MDVRP, avec un nombre de clients entre 50 à 360 et un nombre de dépôts varie entre 2 et 9.

Dans Sambunthan et Kachitvichayanukul [Sambunthan et Kachitvichayanukul, 2010], le PSO amélioré est utilisé pour résoudre MDVRP avec mutation et inertie améliorée.

[Escobar, 2013] a résoudre le MDVRP a l'aide de l'algorithme granular tabu search (GTS) qui basé sur l'idée bien connue de recherche granulaire introduite par Toth et Vigo en 2003 [Toth et Vigo, 2003].

[Tu et al., 2014] ont proposé une méta-heuristique basées sur un diagramme de voronoi à deux niveaux appelée BVDH proposée par les auteurs pour résoudre des instances à grand échelle du MDVRP. Le diagramme de Voronoi à deux niveaux se compose de K-ring Voronoi voisins d'un nœud et d'une ligne. Son niveau supérieur, dérivé des dépôts, permet d'affecter les clients aux dépôts. Le niveau inférieur du diagramme, qui est dérivé des nœuds clients, limite l'espace de recherche de réaffectation des clients entre les dépôts et de réorganisation des clients entre les itinéraires de chaque dépôt à ses voisins de Voronoi. Les auteurs ont démontré les performances de BVDH sur 33 instances de référence MDVRP est sept instances réelles dans lesquelles le nombre de dépôt varie entre 22 et 20 tandis que le nombre de clients varie de 2000 à 20000.

Une autre MDVRP qui lié aux livraisons et ramassages simultanées a été proposé par les auteurs de [j.Li et al., 2015] ont résolu en utilisant une approche méta-heuristique basé sur une recherche locale itérée. Il a été observé à partir des résultats de calcul testés que l'approche proposé surpasse les méthodes précédentes et elle est meilleure que l'utilisation de la recherche

de grands voisinages, l'optimisation des essaims de particules et l'optimisation des colonies de fourmis.

Plus récemment, [Oliveira et al., 2016] ont introduit le premier algorithme Coévolutif coopératif pour aborder le MDVRP. Dans cet algorithme génétique parallèle, chaque dépôt et ses clients assignés sont représentés comme individu d'une population différente. L'affectation client-dépôt est basée sur la distance d'un nœud client par rapport aux dépôts et sa distance par rapport au nœud voisin le plus proche. Les auteurs ont forgé le nom de « Coévolutionnaire » puisque chaque population évolue séparément et simultanément, mais la qualité d'un individu dépend de sa capacité à coopérer avec des solutions à dépôt unique d'autre population pour former une solution complète supérieur au problème global. Les auteurs ont mesuré les performances de l'algorithme sur 33 instances de référence MDVRP standard, où l'algorithme Coévolutif coopératif était considérablement réduit.

Toujours dans la même année [Shuihua Wang et al., 2016] ont proposé un nouvel algorithme génétique adaptatif de mise à l'échelle de la condition physique avec recherche locale (FISAGARLS) pour résoudre le MDVRP. Cette technique permit de convertit la valeur de condition physique Brute en une nouvelle adaptée à la sélection. La stratégie des taux adaptatifs modifie les probabilités de croissements et de mutation en fonction de la valeur de fitness. Le mécanisme de recherche locale exploite l'espace d'un problème de manière plus efficace. Les auteurs ont utilisé 33 problèmes de référence MDVRP avec un nombre de client varie entre 48 et 360 et un nombre de dépôts varie entre 2 et 9.

Les recherches de Kaabachi , Jriji et Krichaen [Kaabachi et al., 2017] ont amélioré l'ACO en ajoutant une recherche locale pour résoudre le MDVRP.

Bezerra, De Souza, and Souza [Bezerra et al., 2018] and Alinaghian et Shokouhi [Alinaghian et Shokouhi, 2018] ont résolu le MDVRP grâce à des variantes de l'algorithme de recherche de voisinage variable qui est une recherche générale de voisinage variable (GVNS) et ont mesuré les performances de leur méthode de résolution uniquement sur les 10 instances de test sans contrainte de durée de tournées.

[Barma et al., 2019] ont développé une méta-heuristique bio-inspirée nommée Discete Antlion Optimization (DALO) qui est complétée par 2-opt pour l'optimisation locale. Les auteurs ont testé DALO exclusivement sur 23 instances de référence MDVRP où les temps de service sont tous nuls.

[Yanjun shi et al., 2020] ont proposé une méthode de résolution heuristique pour résoudre le problème de collecte des déchets dans lequel les déchets ménagers et solides urbains sont amenés des point de collecte des déchets aux usines d'élimination des déchets. La collecte des déchets à partir des points de collecte est modélisé comme un problème de tournées de véhicules multi-dépôts (MDVRP). Visant à minimiser la distance totale de transport. Dans cette méthode, les auteurs attribuée d'abord les points de collecte des déchets aux usines d'élimination des déchets en fonction de la distance la plus proche. Puis, chaque usine résout respectives le problème de tournées d'un seul véhicule (VRP), en affectant les clients aux véhicules et en planifiant l'ordre dans lequel les clients sont visités par les véhicules. Dans la dernière étape ont proposé l'algorithme d'optimisation de combinaison de secteurs (SCO) pour générer

plusieurs solutions initiales, puis ces solutions initiales sont améliorées en utilisant la stratégie merge-head and drop-tail (MHDT). Après un certain nombre d'itération, la solution optimale de la dernière génération est apportée. Les auteurs ont démontré les performances de l'algorithme MHDT sur 23 instances de référence MDVRP avec un nombre de clients varie entre 50 et 360 tandis que le nombre de dépôts varie entre 2 et 9.

Les méta-heuristiques ont été découvertes comme la méthode la plus efficace pour résoudre de nombreux problèmes d'optimisation difficiles. 42% des travaux de recherche sur MDVRP utilisent la méthode méta-heuristique pour résoudre le problème, 33% ont utilisé l'heuristique et 20% ont utilisé la méthode exacte pour les études récentes. D'autre part, les méta-heuristiques à la capacité de comprendre efficacement l'espace de solution qui évite de tomber dans les optima locaux et les optima globaux. C'est pour cette raison, les Méta-heuristiques sont souvent utilisées pour résoudre les problèmes MDVRP.

2.6 Discussion et comparaison

D'après l'étude de ces méthodes, on peut dire qu'il n'existe pas une approche meilleure que l'autre pour résoudre le problème, et que chaque méthode a des avantages et des limites.

Le tableau suivant synthétise quelque méthode proposée dans la littérature et les principales caractéristique des algorithmes pour MDVRP. Tel que, « c » représente le nombre de clients et « d » représente le nombre de dépôts.

Tableau2.2 : Tableau synthétise des méthodes exactes pour MDVRP

Les méthodes exactes			
Auteurs	**Type d'algorithme**	**Taille d'instances résolues**	
		c	d
L'aporte, Nobert et Aspin (1984)	Séparation et évaluation progressive	C=40	d≤8
L'aporte, Nobert et Taillerfer (1988)	Transformation de graphe. Résolution de problème d'affectation par séparation et évaluation progressive	**c**≤80	**d**≤3
Toth et al (2002)	Borne inferieur à l'aide d'une méthode de programmation linéaire en nombre entiers (PLNE)		
Baldacci et Minzoggi (2009)	La formulation de partitionnement d'ensembles	c≤200	2≤d≤5
Contrado et Martinelli (2014)	La fixation de variables, La génération de colonnes et de coupes et l'énumération de colonnes	50≤c≤249	2≤d≤6

Tableau2.3 : Tableau synthétise des méthodes heuristiques pour MDVRP

Les heuristiques			
Auteurs	**Type d'algorithme**	**Taille d'instances résolues**	
		c	**d**
Tellman (1969)	Méthode des gains : Construction de tournées aller-retour, fusions de tournées	c=6	d=2
Tellman et Heiring (1971)	Utilisation de l'algorithme de Tellman (1969), Evaluation des fusions à l'intérieur d'un arbre de décision	$10 \leq c \leq 50$	$3 \leq d \leq 5$
Tellman et Cain (1972)	Utilisation de l'algorithme de Tellman(1969) Evaluation des fusions dans un arbre d'énumération	$10 \leq c \leq 50$	$d \leq 5$
Wren et Holiday (1972)	Construction d'une solution initiale par une méthode de balayage. Amélioration (mouvements simples, échange de clients, etc…)	$21 \leq c \leq 183$	$d \leq 2$
Gillett et Jonhson (1976)	Création de regroupements de clients autour des dépôts. Résolution de PTV	$50 \leq c \leq 249$	$2 \leq d \leq 5$
Chao, Golden et Wasil (1993)	Méthode des registres. Raffinements	$50 \leq c \leq 360$	$2 \leq d \leq 9$
Golden, Mangnanti et Nguyan (1977)	Affectation des clients aux différents dépôts.	c=600	d=2
Raft (1982)	Construction de tournées. Affectation des tournées aux dépôts.	$50 \leq c \leq 249$	$d \leq 5$
Salhi et Sari (1977)	Une extension du mix flotte au MDVRP, plusieurs types d'échanges (inter-dépôts, intra-dépôts).	C= 360	$2 \leq d \leq 9$
Wang Xingyin, Bruce Golden, Edward Wasil et Rui Zhang (2016)	Le min-max Split Delivery MDVRP with minimum Service Time Requirement (min-max SDMDVRP-MDTR)		

Tableau 2.4: Tableau synthétise des méthodes méta-heuristiques pour MDVRP

Les méta-heuristiques			
Auteurs	**Type d'algorithme**	**Taille d'instances résolues**	
		c	**d**
Renaud, Laporte et Boctor (1996)	Recherche avec tabou avec intensification et diversification	50≤c≤360	2≤d≤9
Beatrice et al (2009)	Utilisation de l'algorithme génétique, un codage indirect, stratégie adaptative d'échange de mutation inter-dépôts	50≤c ≤360	2≤d≤9
Sumbunthan et Kachitvichayanukul (2010)	Optimisation d'essaim de particules avec plusieurs structures d'apprentissage social	50≤c≤249	2≤d≤5
Escobar (2013)	Granular Tabu Search	50≤c≤360	2≤d≤9
Tu et al (2014)	Diagramme de voronnoi à deux niveaux appelée BVDH	2000≤c≤20000	20≤d≤22
Ji.L et al (2015)	La recherche locale itérée	50≤c≤249	2≤d≤5
Oliveira et al (2016)	Co-évolutif coopératif. Affectation clients-dépôts	48≤c≤360	2≤d≤9
Kaabachi, Jriji et Krichan (2017)	Optimisation amélioré de colonie de fourmis (IACO) avec recherche locale	48≤c≤360	2≤d≤9
Bezerra, Desouza et (2018)	Une recherche recherche générale de voisinage variable (GVNS)	50≤c≤360	2≤d≤9
Berma et al (2019)	Discrete Antolion Optomisation (DALO) par 2-opt.	50≤c≤360	2≤d≤5
Yunjun shi, Lingling Lv, Fanyi Hu et Qiamei Han (2020)	La distance la plus proche. Résolution de VRP. Optimisation de combinaison de secteurs (SCO). Stratégie merge-head and drop-tail (MHDT)	50≤d≤360	2≤d≤9

L'analyse que nous verrons de faire permet de voir jusqu'à quel point le MDVRP est une variante importante du VRP et quelles sont les méthodes exactes et heuristiques qui ont été développées afin de résoudre.

2.7 Conclusion

Comme nous avons pu le voir tout au long de ce chapitre, le MDVRP n'est pas une taché facile, il est de type NP-difficile. La considération de plusieurs dépôts rend le problème plus difficile que le VRP classique. De plus, il est intéressant d'intégrer des contraintes de capacité à la fois sur les véhicules et les dépôts de manière à se rapprocher au maximum des problèmes réalistes.

Dans ce chapitre dans un premier temps, nous avons défini le problème de tournées de véhicules classique (VRP), puis nous avons identifié ces variantes. D'autre coté nous avons parlé sur le problème de tournées de véhicules multi-dépôts et nous avons donné sa définition et sa formulation mathématique. Ensuite, nous avons également présenté les différentes approches de résolution de MDVRP comme les méthodes exactes, les heuristiques et les méta-heuristiques, et nous avons donné des exemples des travaux développés pour résoudre ce problème.

Dans le chapitre suivant, notre contribution sera présentée pour résoudre le problème de tournées de véhicules multi-dépôts.

Partie 2

Contribution et Implémentation

Chapitre 3

Conception et Contribution

3.1 Introduction

Les algorithmes bio-inspirés sont une branche prometteuse des méthodes de résolution des problèmes d'optimisation. L'algorithme inspiré des réactions chimique (CRO) est une méta-heuristique récemment établie pour l'optimisation, inspirée par la nature des réactions chimiques. Une réaction chimique est un processus naturel de transformation des substances instables en stables, [Lam et Li, 2012].

Vu les limites rencontrées par les approches exactes de trouver des solutions optimales dans un temps raisonnable pour les grandes instances, nous avons opté pour les méta-heuristiques basées population pour remédier à ce problème. L'algorithme inspiré des réactions chimiques CRO (Voir chapitre 1) est proposé pour résoudre le MDVRP.

Dans ce chapitre, nous décrirons d'abord notre approche proposée, son processus et ses principales caractéristiques. Ensuite, nous discuterons de son application pour le MDVRP et comment adapter des opérateurs à ce problème. Un résumé de tout cela sera donné plus tard.

3.2 Motivation

CRO est une méta-heuristique à base population. Cet algorithme est une technique d'optimisation récemment développée qui s'inspire du fonctionnement des réactions chimiques. Il simule les itérations de molécules en créant des réactions élémentaires qui se combinent de façon aléatoire. Cela permet aux molécules d'explorer l'espace de recherche des solutions afin de trouver le meilleur minimum global.

Malgré qu'il est un algorithme relativement nouveau, CRO a été appliqué pour la résolution des différents problèmes d'optimisations combinatoires complexes mono-objectifs et multi-objectifs comme le problème de voyageur de commerce TSP, le problème de sac à dos …etc. Les résultats trouvés montrent bien l'efficacité de cette méthode.

Des comparaisons ont montré que CRO atteint des performances très compétitives par rapport aux d'autres approches évolutives ces derniers années. [Lam et Li, 2012].

La principale différence entre l'algorithme CRO et les autres techniques évolutives est que la taille de la population dans le CRO peut changer après chaque itération durant l'exécution, alors que dans toutes les autres techniques, la taille de la population est généralement fixe et inchangée. Cela donne une grande flexibilité au CRO pour la résolution des différents problèmes.

Cet algorithme puissant est caractérisé par trois points forts : premièrement, il combine à la fois les avantages du recuit simulé (SA) c'est-à-dire permet d'accepter des mauvaises solutions qui peut mener la recherche vers la meilleure solution (L'optimum global) [Kirkpatrick et al., 1983] et de l'algorithme génétique (GA) c'est-à-dire l'idée des opérateurs de mutation et de croissement [Goldberg, 1989]. Deuxièmement, il est facile de l'adapter pour traiter des problèmes d'optimisation discrets et n'a pas besoin d'utiliser des techniques de discrétisation. Troisièmement, la taille de la population est variable, ce qui donne au CRO plus de capacité d'adaptation avec un mélange d'intensification qui concentre sur le test du voisinage d'une solution élue et de diversification qui encourage le processus de recherche a examiner des

régions non visitées et à découvrir des solutions différents des solutions rencontrées dans des points divers. Et aussi il maintient un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche à travers les opérateurs chimiques et les conditions de conservation de l'énergie.[Lam et Li, 2012].

D'après une bonne recherche dans la littérature, l'originalité de notre travail réside dans l'utilisation de l'algorithme CRO pour la première fois afin de résoudre MDVRP, nous avons proposé quatre opérateurs adaptés au problème. Nous avons choisi d'appliquer l'algorithme CRO afin de montrer leur bonne adaptation et leur efficacité de résoudre les problèmes d'optimisation de la classe NP-difficile.

3.3 Présentation de notre MDVRP

Nous avons établi dans les chapitres précédents la structure générale du problème MDVRP, tel que ce problème est classé parmi les problèmes d'optimisation mono-objectif avec contraintes, de Forme Linéaire sur Landscape Multimodale et les Variables de type discrètes déterministe c'est-à-dire :

Les demandes des clients sont du type entier où chaque demande de client ne dépasse pas une capacité spécifique pour les véhicules et tous les véhicules ont la même capacité. Lorsque le véhicule atteint sa capacité maximale, un autre véhicule est ouvert pour les demandes des clients.

Les demandes des clients sont prises à partir des dépôts qui ont également une capacité égale à la capacité d'un véhicule multipliée par le nombre de véhicules de dépôt et tous les dépôts ont le même nombre de véhicules et la même capacité. Chaque fois qu'un dépôt s'ouvre pour les demandes des clients jusqu'à ce qu'il atteigne sa capacité maximale, un autre dépôt est ouvert jusqu'à ce que toutes les demandes des clients soient terminées.

Comme mentionné précédemment, nous visons à servir tous les clients en minimisant le cout total tout en respectant les contraintes de capacité (véhicule et dépôt), pour y parvenir, nous devons calculer le coût total à l'aide d'une fonction objectif comme suit :

Nous avons un ensemble des clients pour chaque client il y a une demande et des coordonnées (x, y), un ensemble de dépôts pour chaque dépôt il y a une capacité et des coordonnées (x, y) et une flotte de véhicules pour effectuer les livraisons.

On calcule d'abord la distance entre les clients, la distance entre les dépôts et les clients ensuite, nous calculons le coût de toutes les tournées de chaque dépôt.

Le coût de chaque tournée est calculé comme suite : la distance entre le dépôt et le premier client, la distance entre chaque clients et la distance entre le dernier client et le dépôt pour obtenir le coût d'une seule tournée, répétons le processus pour chaque tournée du dépôt. Après, nous avons faire la somme des coûts de toutes les tournées de dépôt pour obtenir le coût d'un dépôt, répétons le processus avec tous les dépôts. Enfin nous sommons le coût de chaque dépôt pour obtenir le coût total.

3.4 Application de CRO pour MDVRP

Le MDVRP étudie dans ce travail est un problème d'optimisation combinatoire de la classe NP-difficile. Au cours de ces années, avec l'évolution des méthodes et les algorithmes d'optimisation développés dans le domaine de recherche opérationnelle plusieurs approches ont été proposées pour résoudre le MDVRP, mais il est difficile de savoir laquelle donne de bons résultats. Cela rendre le domaine de recherche très actif.

Le choix de l'algorithme et de son paramétrage joue un rôle primordial dans la détermination de la qualité des résultats. Malheureusement, il existe très peu de règles universelles permettant de connaitre le paramétrage idéal et généralement seuls l'expertise et l'expérience de l'utilisateur peuvent conduire un bon choix. [Mathieu, 2008].

Dans cette section, nous allons présenter notre approche proposée. Premièrement, nous décrirons le schéma général et le processus de CRO, Ensuite, nous présentons notre solution proposée et les différents opérateurs chimiques proposés.

3.4.1 La représentation de la solution

Une solution au problème MDVRP doit déterminer les dépôts, les clients et les véhicules à affecter à chaque dépôt et les routes à construire pour répondre à la demande des clients avec le coût global minimum tout en respectant les contraintes de capacités.

Le codage est l'une des étapes les plus cruciaux. En effet, la qualité des résultats peut en dépendre complètement. C'est pour cette raison, il faut définir le codage le plus adéquat vis-à-vis du problème à résoudre. Nous proposons de représenter chaque solution par une molécule *M*. Cette molécule à une structure moléculaire $\boldsymbol{\omega}$ qui est représenté par une matrice à deux dimensions. La première colonne représente les dépôts, la deuxième colonne contient les séquences (tournées) selon laquelle les clients doivent être visités par chaque véhicule. Les tournées sont séparées en utilisant le délimiteur '0', les séquences de chaque ligne sont placées dans un 'Vector'. En conséquence, chaque ligne représente les tournées de véhicules correspond à chaque dépôts.

Nous avons choisi de représenter notre solution dans une matrice contrairement à une liste ou d'autre façon car elle fournit un aperçu rapide de la solution. De plus, cela facilite le travail avec les opérateurs utilisés dans l'algorithme CRO.

Exemple : un exemple de test dans lequel on a travaillé contient 50 clients, 4 véhicules avec une capacité égale à 80 et 4 dépôts où chaque dépôt à une capacité égale à 360 (la somme des capacités des véhicules). Une solution faisable et la représentation matricielle correspondante sont représentées dans les figure 3.1 et 3.2. Cette solution utilise seulement trois dépôts. Le dernier dépôt reste fermé car les trois dépôts satisfaire tous les demandes des clients. Donc, on n'a pas besoin d'ouvrir le dernier dépôt.

Remarque : Le 0 représente un délimiteur entre chaque tournée.

Tournée délimiteur

[2]	[16,23,8,5,0,10,45,9,21,39,44,0,12,11,6,0,17,36,49,27,4,47,0]
[1]	[26,7,32,1,41,29,0,20,30,15,0,48,3,24,19,34,0,2,46,28,31,37,43 ,0]
[3]	[18,38,40,0,33,22,13,0,50,25,35,42,0,14]
[4]	null

Figure 3.1 : La représentation matricielle de la solution

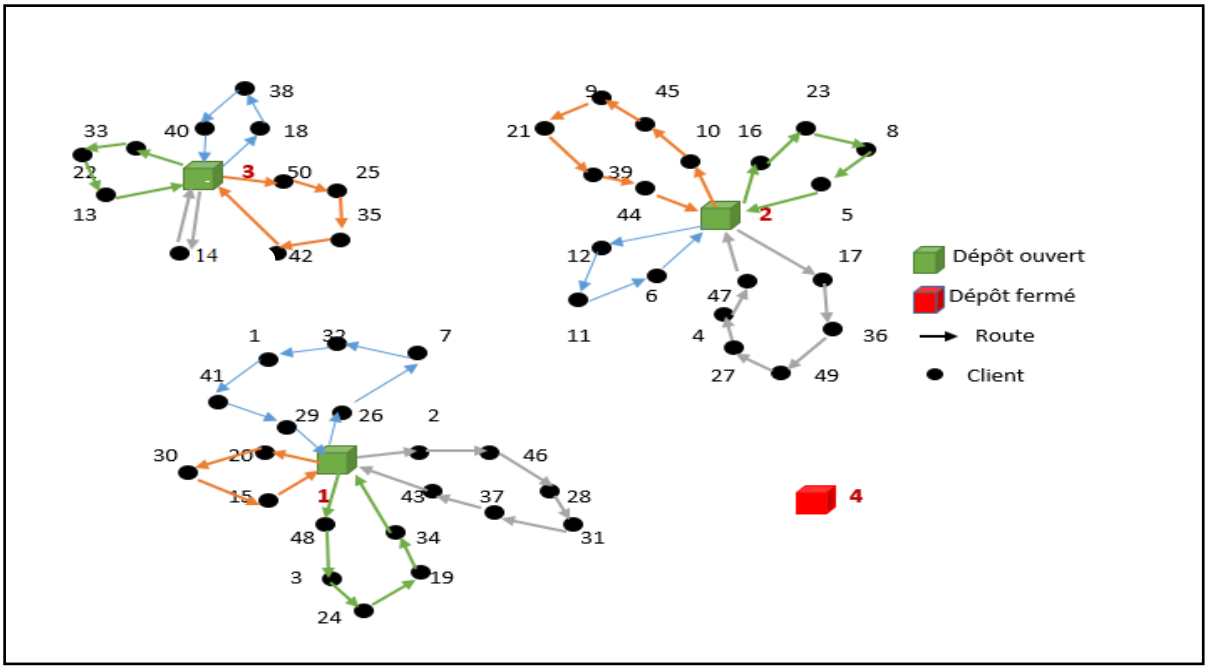

Figure 3.2 : La représentation de la solution

3.4.2 Diagramme de classe pour CRO

La figure suivante représente un diagramme de classes qui modélise les différentes classes de l'algorithme CRO.

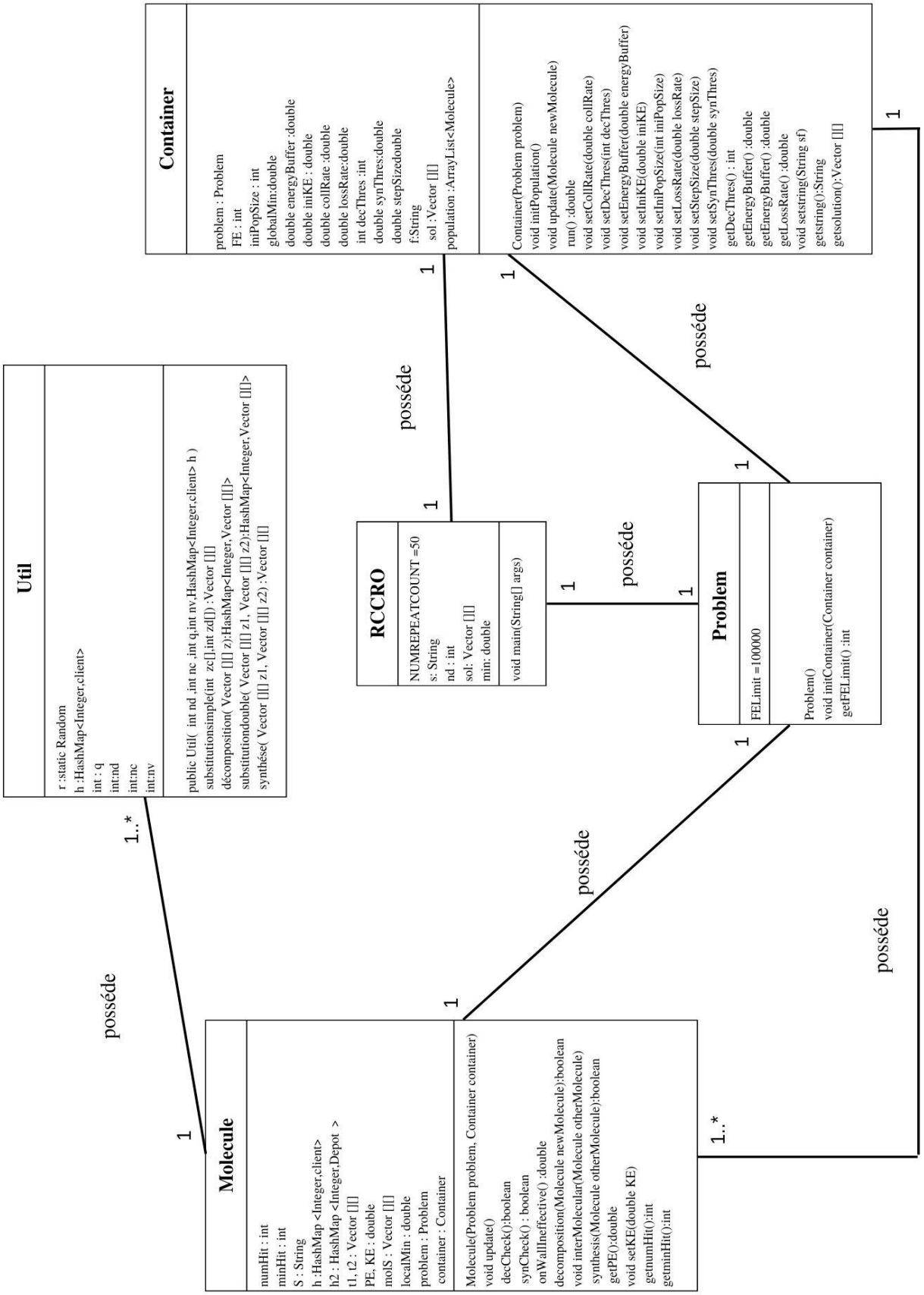

Figure 3.3 : Digramme de classe pour CRO

3.4.3 Les opérateurs chimiques

Comme mentionnée précédemment, CRO prend en charge deux type de collision moléculaires : la collision uni-moléculaires lorsqu'une molécule entre en collision avec la paroi de conteneur, et la collision inter-moléculaire lorsque plusieurs molécules entrent en collision les unes avec les autres. Les collisions uni-moléculaires entrainent une substitution simple et une décomposition. Par contre les collisions inter-moléculaires représentent une substitution double et synthèse. Dans cette section, nous allons détailler chaque type de réaction et leur adaptation selon la représentation de la solution utilisée.

3.4.3.1 La substitution simple

Lorsqu'une molécule entre en collision avec la paroi du conteneur et rebondit en une seule unité, cela s'appelle une substitution simple. Dans cette situation, seule la molécule existan $\boldsymbol{\omega}$ était perturbée en $\boldsymbol{\omega}'$, [Lam et Li, 2012a]. Dans cet opérateur nous avons créé une tournée géante, en transformant toutes les tournées de la solution en une seule. Premièrement, nous avons choisi aléatoirement deux positions différents (P_1, P_2) dans $\boldsymbol{\omega}$ et échangé les deux clients ayants respectivement les positions P_1, P_2 pour obtenir une solution $\boldsymbol{\omega}'$ à partir de $\boldsymbol{\omega}$. Tout en respectant les contraintes de capacité qui signifie que la somme des demandes des clients regroupées en une tournées ne doit pas excéder la capacité de véhicule, sachant que la demande de chaque client ajouté à chaque tournée est inférieur à la capacité de véhicule, et que tous les véhicules sont ici de même capacité. Tous les véhicules affectées à chaque dépôt ne doit pas dépasser la capacité de ce dépôt. Comme montre la figure 3.4 :

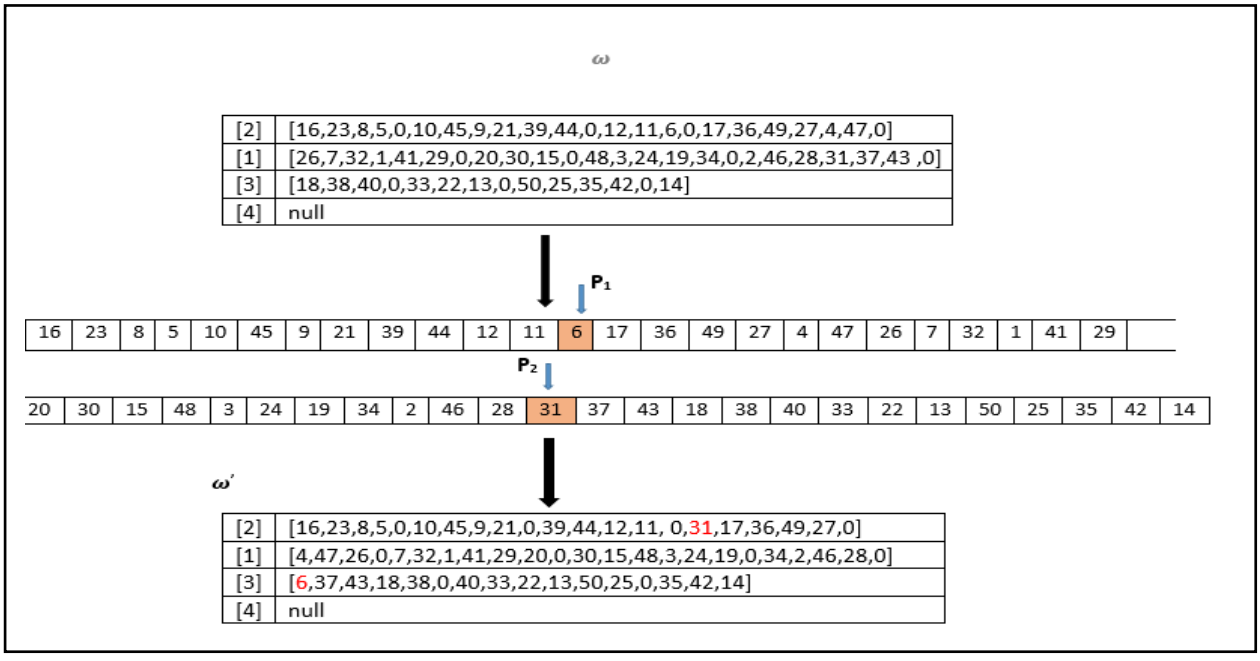

Figure 3.4 : Opérateur de la substitution simple de CRO-MDVRP

Le pseudo-code de l'opérateur de substitution simple est présenté comme suite :

Opérateur de substitution simple

Input M
Output M
Obtenir la structure moléculaire ω de M
Créer une tournée géante à partir des tournées de véhicules existées.
Choisir au hasard deux positions P_1, P_2 dans ω
Echanger les deux clients ayant respectivement les positions P_1, P_2
Obtenir ω' et Respecter les contraintes de capacité de dépôts et de véhicules

Si $PE(\omega) + KE_\omega \geq PE(\omega')$ **Alors**
Accepter la nouvelle molécule M (ω ');
Remplacer M (ω) par M (ω');
Sinon
Rejeter M (ω');
FinSi ;

3.4.3.2 La décomposition

D'après [Lam et Li, 2012a], la décomposition signifie qu'une molécule $M\boldsymbol{\omega}$ frappe la paroi du conteneur puis se divise en deux parties $M\omega_1'$ et $M\omega_2'$. Dans cet opérateur, pour générer deux solutions ω_1' et ω_2' à partir de la solution ω , on choisit une position P aléatoirement de ω, la nouvelle solution ω_1' prend la même partie [0, P] de ω, la même chose pour ω_2'elle prend la partie [P, size (ω)] de, les parties restante de ω_1' et ω_2' est généré aléatoirement. En se basant sur la capacité de véhicule, où la demande de chaque client ajouté à chaque tournée est inférieure à la capacité de véhicule, et tous les véhicules affectés à chaque dépôt ne doit pas dépasser la capacité de ce dépôt. Comme montre la figure 3.5 :

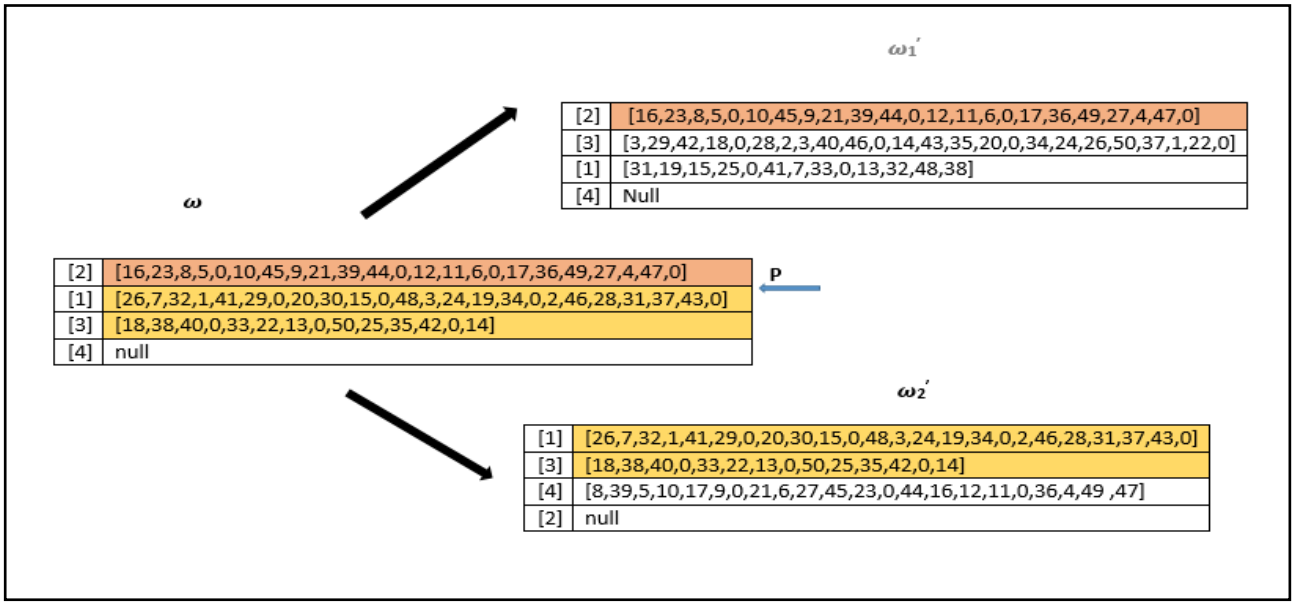

Figure 3.5 : Opérateur de la décomposition de CRO-MDVRP

Le pseudo-code de l'opérateur de décomposition est présenté comme suite :

Opérateur de décomposition

Input M, Buffer
Output *Mω1', M ω2', Buffer*
Obtenir la structure moléculaire ω de M
Choisir une position aléatoire *P* dans *ω*
ω1', ω2' prennent la même partie [0, *P*], [*P*, size(*ω*)] de *ω* respectivement
La partie restante de *ω1', ω2'* est générée aléatoirement.
Calculer temp1= *PE ω+ KE ω -PE ω1' - PE ω2'*
Si (condition de conservation d'énergie de décomposition est satisfaite) **Alors**
Générer $\delta_1 et\ \delta_2 \in [0, 1]$
Buffer = Buffer × (1 − $\delta_1\delta_2$)
Si $PE(\omega) + KE_\omega + \delta_1\delta_2$ buffer $\geq PE(\omega_1') + PE(\omega_2')$**Alors**
Remplacer M (ω) par M (ω1') et M (ω2') ;
Mise à jour du Buffer;
Sinon
Rejeter M (ω1') + M (ω2') ;
FinSi ;

3.4.3.3 La substitution double

La substitution double se produit lorsque deux molécules entrent en collision et forment ensuite deux nouvelle molécules. L'effet du changement d'énergie des molécules est similaire à celle d'une substitution simple. [Lam et Li, 2012a]. L'idée de cet opérateur et de générer deux solutions ω_1' et ω_2' à partir de ω_1 et ω_2. Pour générer ces deux solutions nous avons choisi deux positions aléatoires, la position $P_1 \in [0, size(\omega_1)-1]$ de ω_1 et $P_2 \in [0, size(\omega_2)-1]$ de ω_2. Ensuite, il copie les éléments entre [0, $P1_1$ de ω_1 et [0, P_2] de ω_2 aux positions correspondantes de ω_1' et ω_2' respectivement. Les parties restantes de ω_1' et ω_2' sont complètent à partir de ω_2 et ω_1 respectivement. Tout en respectant les contraintes de capacité qui signifie que la somme des demandes des clients regroupées en une tournée ne doit pas excéder la capacité de véhicule, sachant que la demande de chaque client ajouté à chaque tournée est inférieur à la capacité de véhicule, et que tous les véhicules sont ici de même capacité. Tous les véhicules affectés à chaque dépôt ne doit pas dépasser la capacité de ce dépôt. Comme montre la Figure 3.6 :

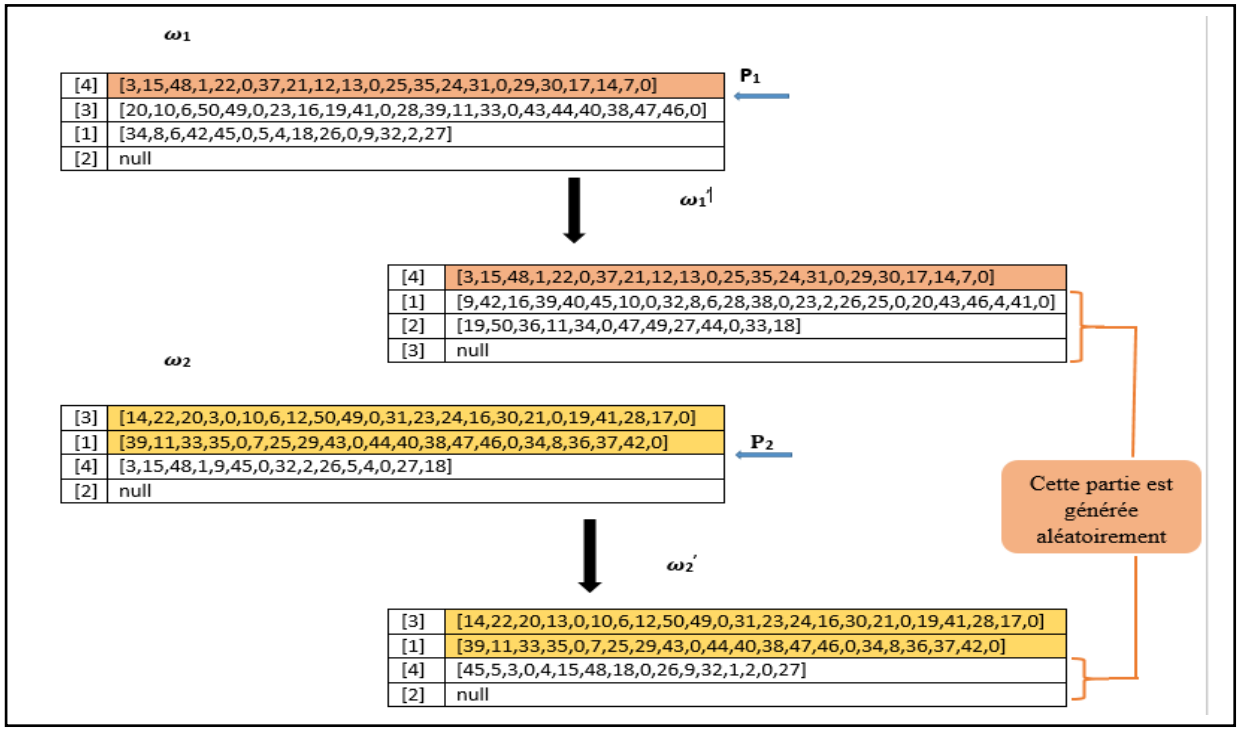

Figure 3.6 : Opérateur de la substitution double de CRO-MDVRP

Le pseudo-code de l'opérateur de substitution double est présenté comme suite :

Opérateur de substitution double

Input M1, M2
Output Mω1', Mω2'
Obtenir les structures moléculaires *ω1* et *ω2* de *Mω1*, *Mω2*
Choisir deux positions aléatoires *P1*, *P2* dans *ω1, ω2*;
ω1' et *ω2'* Prennent la mêmes parties [0, *P1*] de *ω1* et la partie [0, *P2*] de *ω2* respectivement
Les parties restantes de *ω1'* et *ω2'* sont générés aléatoirement à partir de *ω1* et *ω2* respectivement

Si $PE_{\omega_1} + KE_{\omega_1} + PE_{\omega_2} + KE_{\omega_2} \geq PE_{\omega_1'} + PE_{\omega_2}$ **Alors**
Remplacer M (ω1) et M (ω2) par M (ω1') et M (ω2') ;
Sinon
Rejeter M (ω1') et M (ω2') ;
FinSi ;

3.4.3.4 La synthèse

La synthèse est le processus dans lequel deux molécules entrent en collision entre eux et donnent une nouvelle molécule. L'idée de synthèse est de permettre au système d'explorer l'espace de recherche comme la décomposition. [Lam et Li, 2012a]. L'intérêt de l'utilisation de cet opérateur et la génération d'une nouvelle solution ω' à partir des deux solutions ω_1 et ω_2.

Nous avons choisi au hasard une position P ∈[0, size(ω_1)-1] de ω_1. Ensuite, ω' prend la première partie de la solution [0, P] à partir de ω_1 et la seconde partie à partir de ω_2. Tout en respectant les contraintes de capacité qui signifie que la somme des demandes des clients regroupées en une tournées ne doit pas dépasser la capacité de véhicule, sachant que la demande de chaque client ajouté à chaque tournée est inférieur à la capacité de véhicule, et que tous les véhicules sont ici de même capacité. Tous les véhicules affectés à chaque dépôt ne doit pas dépasser la capacité de ce dépôt. Comme montre la Figure 3.7 :

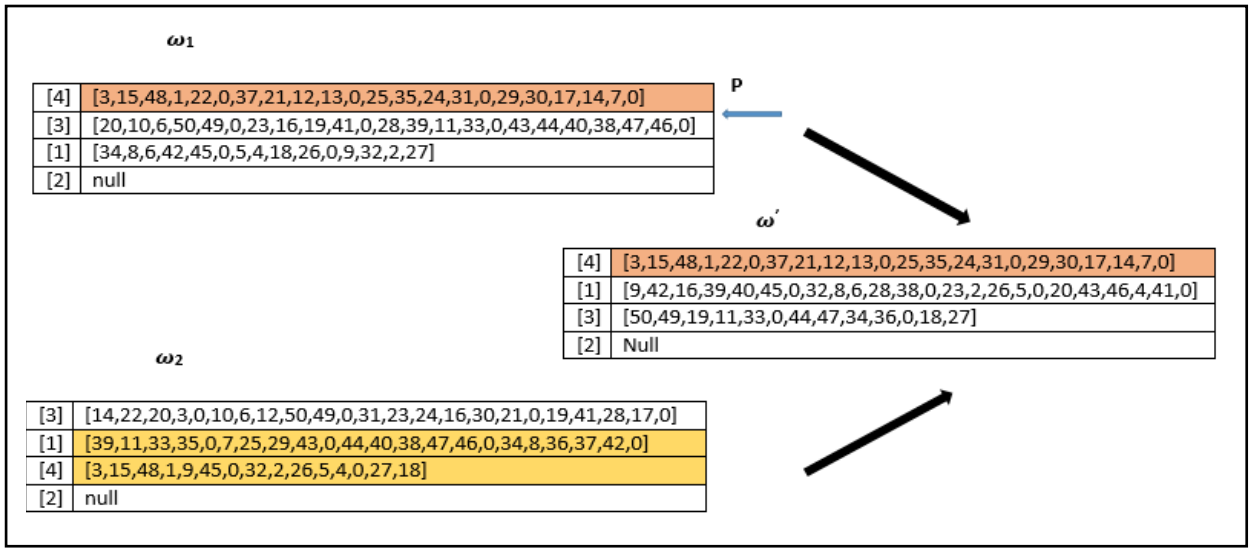

Figure 3.7 : Opérateur de synthèse de CRO-MDVRP

Le pseudo-code de l'opérateur de synthèse est présenté comme suite :

Opérateur de synthèse

Input M1, M2
Output Mω'
Obtenir les structures moléculaires *ω1* et *ω2* de *M1, M2*
Choisir une position aléatoire *P* [0, size (ω1)-1]
ω' ← Prend la même partie de la solution [0, *P*] à partir de *ω1* et la seconde partie à partir de *ω2*.
Si $PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'}$ **Alors**
M (ω1) et M (ω2) sont remplacées par M (ω');
Sinon
Rejeter M (ω');
FinSi ;

3.5. Représentation générale de l'algorithme CRO

L'algorithme inspiré des réactions chimiques s'est avéré être un outil efficace pour résoudre des problèmes d'optimisation combinatoire. CRO est une méta-heuristique basée sur le processus naturel de transformation de substances instables en stables. [Lam et Li., 2012].

Dans CRO chaque molécule *M* a plusieurs attributs, les attributs essentielles comprennent : une structure moléculaire ω, une énergie potentielle (*PE*) et l'énergie cinétique (*KE*), et d'autre attributs facultatifs adoptés dans la plupart des variantes CRO publiées sont : le nombre de collision que la molécule subit *NumHit*, la meilleure valeur que *M* a eu durant son existence *MinPE* et nombre de collision associé à *MinPE* qui est *MinHit*. ω, représente une solution d'un problème donnée, *PE* définit la valeur de la fonction objective de ω tandis que *KE* quantifie la tolérance du système d'accepter une mauvaise solution. [Sun et al., 2011].

Pour résoudre le MDVRP, nous avons adapté l'algorithme CRO illustré à la figure 3.8. Le processus CRO proposé comprend trois étapes : l'initialisation, les itérations et la phase finale.

Dans la phase initialisation un ensemble de molécule N est généré aléatoirement. De plus les valeurs sont attribut aux paramètres CRO.

Après ça on passe à la phase d'itération nous déterminons d'abord s'il s'agir d'une réaction unimoléculaire ou intermoléculaire en comparant un nombre aléatoire r [0,1].

Si r>collRate alors une réaction unimoléculaire sera produite dans cette étape si la condition de décomposition est vérifiée alors une décomposition sera appliqué. Sinon, une substitution simple sera choisie.

Dans le cas où on a une collision intermoléculaire, se la condition de synthèse est satisfaite alors une synthèse sera appliquer sinon une substitution double sera choisie.

Finalement, la population est mise à jour, le processus est répété jusqu'à ce que le critère d'arrêt soit atteint. Dans notre cas on a choisi le nombre maximal d'itération effectué comme critère d'arrêt. Comme montre la figure 3.8 illustré un schéma général de processus CRO-MDVRP.

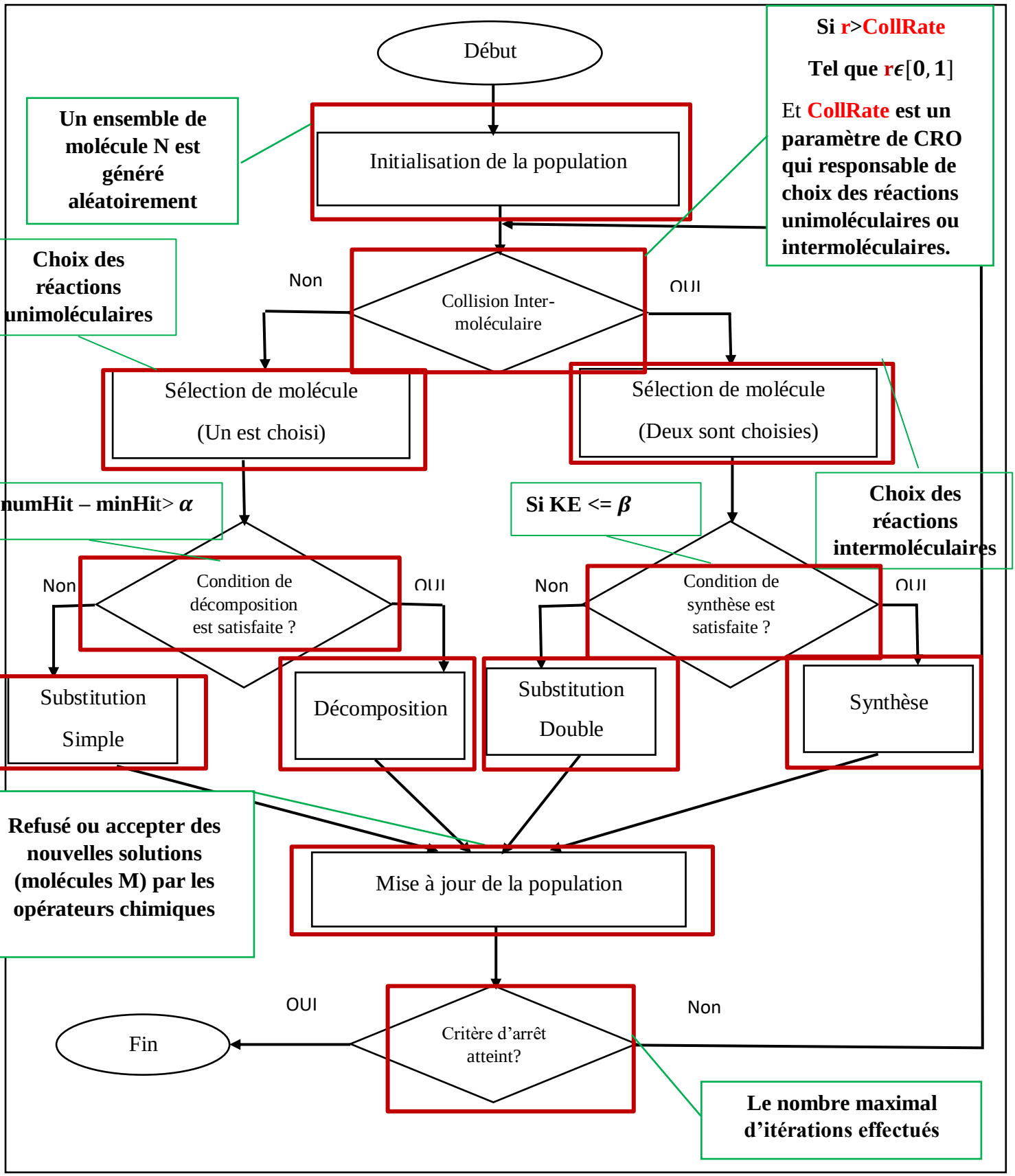

Figure 3.8 : Schéma général de l'algorithme CRO-MDVRP

3.6 Conclusion

Dans ce chapitre, on a présenté notre contribution pour résoudre MDVRP basée sur la méta-heuristique inspirée des réactions chimiques CRO. CRO est un outil puissant pour résoudre les problèmes d'optimisation de la classe NP-difficile.

Tout d'abord, nous avons présenté l'adaptation de l'algorithme CRO pour résoudre le problème MDVRP. Ensuite, nous avons présenté ses étapes et son processus. Nous avons proposé quatre opérateurs chimiques, à chaque itération de l'algorithme l'un de ces operateurs est appliqué selon certaines conditions.

Dans le prochain chapitre, nous allons faire une étude expérimentale et nous discuterons les résultats obtenus.

Chapitre 4

Implémentation et tests

4.1 Introduction

Après avoir présenté notre contribution en détails dans le chapitre précédent, ce chapitre est consacré à l'étude expérimentale des résultats trouvés. Nous commençons par décrire les outils et le matériel utilisés. Ensuite, nous discutons les résultats de l'application de l'algorithme CRO pour résoudre le MDVRP. Afin de tester la performance l'approche que nous avons proposée, nous l'avons testée et validée sur un ensemble de tests provenant de la littérature. Les études expérimentales faites ont montré l'efficacité des méthodes proposées de trouver des solutions de bonne qualité pour le problème étudiée dans ce travail.

4.2 Environnement de développement

Pour mener à bien notre projet, nous devons être conscients de deux facteurs importants : l'environnement matériel et l'environnement logiciel.

Tableau 4.1 : Environnement de développement

	Outils et matériel	**Description**
Environnement matériel	Marque	Lenovo
	Processeur	AMD A4
	Mémoire installé(RAM)	4,00 GO
	Système d'exploitation	Windows 10 professionnel 64 bits
Environnement logiciel	Eclipse	Eclipse est un environnement de développement (IDE), il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation. Permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation, Dont le C|C++ et PHP. Eclipse IDE est principalement écrit en java, il nécessite une machine virtuelle java (JRE) pour fonctionner mais pour compiler du code java, un kit de développement (JDK) est indispensable. Récupéré sur www.eclipse.org/ecmilseide
Langage de programmation	Java	Java est un langage de programmation, est une plateforme informatique qui a été créé par Sun Microsystems en 1995. Beaucoup d'application et des sites web ne fonctionnent pas si java n'est pas installé et leur nombre ne cesse de croitre chaque jour. Java est rapide, sécurisé et fiable. Des ordinateurs portables aux centres de données, des consoles de jeux aux superordinateurs Scientifique, des téléphones portables à internet, la technologie java est présente sur tous les fronts. Il est fourni avec un ensemble d'outils (le JDK java développement kit) est un ensemble de package : ensemble de classes. Ces différentes classes de base couvrent

		beaucoup de domaine (entrée/sortie, interface graphique, réseau, etc.). Cette richesse en « Librairies standards » explique surement en partie le succès de java. Le langage lui-même se trouve dans le package java. Lang. Java est donc : ✓ Un langage de programmation objets. ✓ Une architecture de machine virtuelle. ✓ Un ensemble d'outils. ✓ Un ensemble de librairies (packages) de base.

4.3 Présentation de l'application

Pour rendre l'utilisation de notre approche aussi simple, nous avons créé une interface graphique simple qui permet de faciliter le travail à n'importe quel utilisateur et permet d'afficher la solution optimale et le coût minimal de cette solution. Pour cela, nous utilisons la bibliothèque Swing sous éclipse.

Swing est une bibliothèque pour le langage de programmation java, elle offre la possibilité de créer des interfaces graphiques quel que soit le système d'exploitation.

La figure 4.1 montre la page d'accueil de notre application, Cette interface donne l'accès à toutes les autres interfaces.

Figure 4.1 : Accès aux fichiers

Le bouton « Choisir le fichier » permet de spécifier le chemin d'accès d'un fichier, lorsqu'on clique sur ce bouton, une fenêtre sera ouverte. Cette fenêtre permet de parcourir les différents répertoires et sous-répertoires de l'ordinateur (voir la figure 4.2)

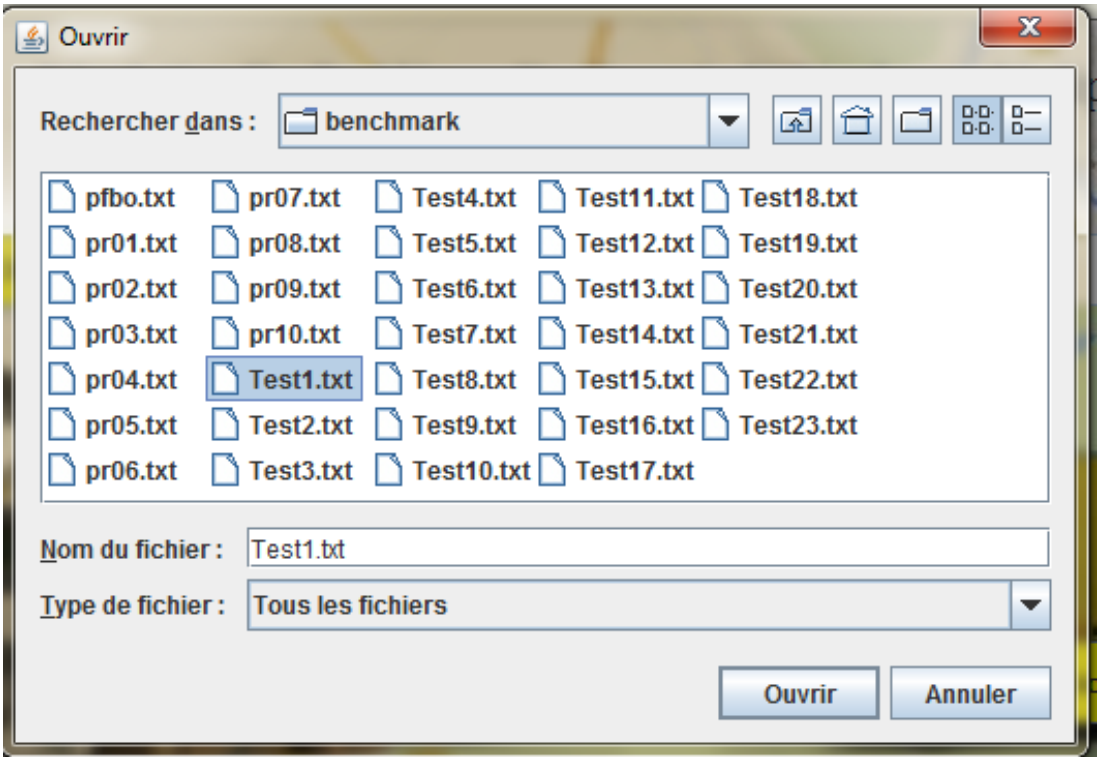

Figure 4.2 : Interface graphique permettant de rechercher les fichiers à traiter

Ensuite, nous cliquant sur le bouton « Accéder » pour passer à la deuxième interface. Comme le montre la figure 4.3 :

Dans le cas où aucun fichier n'est choisi, le message « Veillez sélectionner un fichier SVP ! » sera affiché.

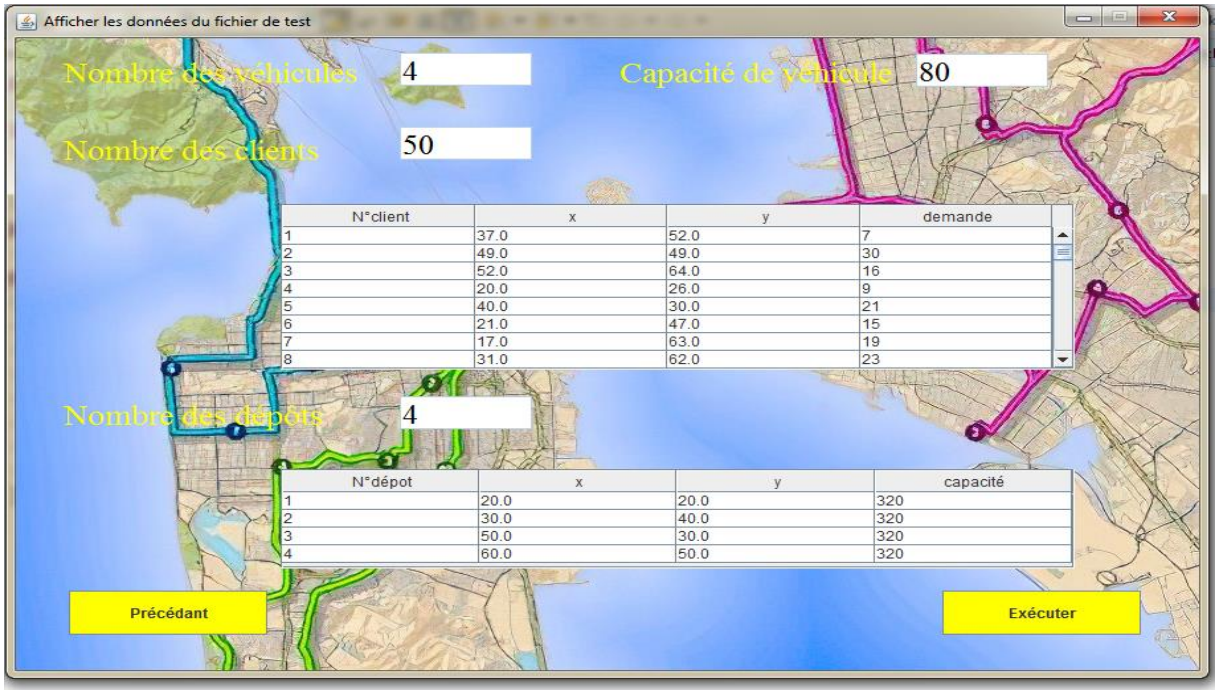

Figure 4.3 : Afficher les données du fichier test

Au niveau de cette interface :

Les données du fichier test choisi sont affichées : le nombre de véhicules et leur capacités, le nombre de clients et un tableau qui contient l'ID, les cordonnées (X, Y), la demande de ces clients et aussi les informations sur les dépôts.

Si en cliquant sur le bouton « Précédant », nous revenons à l'interface « Accès aux fichiers » et nous pouvons choisir un autre fichier. Le bouton « Exécuter» permet de lancer l'exécution de l'algorithme CRO, une fois l'exécution terminée une troisième interface sera affiché (voir la figures 4.4)

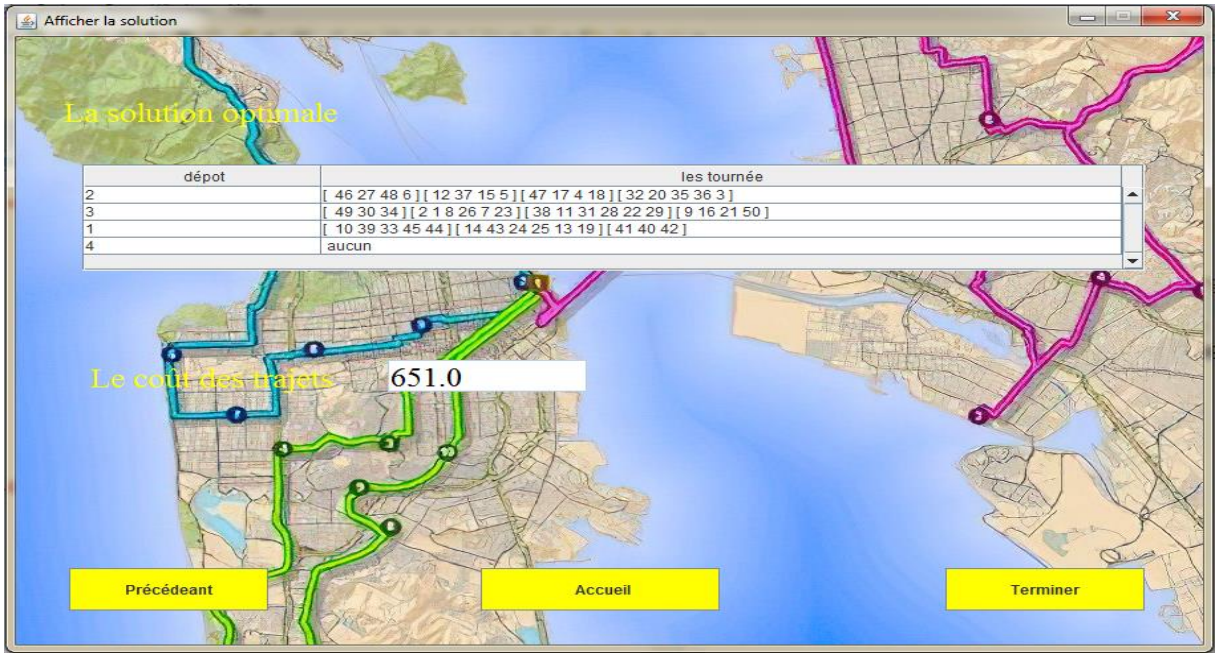

Figure 4.4 : La solution trouvée au problème MDVRP

Au niveau de cette interface : la solution optimale trouvée et le coût minimal de cette solution seront affichées.

Si nous voulons revenir à la première interface pour choisi un autre fichier en cliquant sur le bouton « Accueil », et pour quitter de l'application en cliquant sur le bouton « Terminer ».

4.4 Etude expérimentale et discussion des résultats

L'idée de cette section est d'évaluer les performances de notre approche. Nous avons testé notre algorithme CRO-MDVRP sur les 33 instances proposées dans la littérature pour le MDVRP. Ensuite, nous avons comparé nos résultats avec celles des autres algorithmes proposés pour résoudre le même problème. Finalement, nous analysons les résultats de chacune des algorithmes sur ces instances et terminerons par une étude comparative des résultats.

4.4.1 Description des instances

Dans ce travail, nous avons utilisé comme ensemble de tests les 33 instances proposées pour le MDVRP qui peuvent être téléchargé à partir de web VRP sur le site web[www.bernabe.dorronsoro.es/vrp/] [Web 9], les instances 1 à 7 ont été introduit par Christofides et Eilon [Christofides et Eilon, 1969], 8 à 11 ont été décrits dans [Gillett et Johnson,

1976], les instances 12 à 23 ont été proposées par Chao et al [Chao et al., 1993]. Enfin les instances 24 à 33 ont été introduites par Cordeau et al [Cordeau et al., 1997]. Chacune de ces 33 instances correspond à un nombre de client varie entre 50 et 360, le nombre de véhicule varie entre 4 et 12 avec une capacité compris entre 60 et 500, un nombre de dépôt varie entre 2 et 9. Les informations de base de toutes les instances sont présentées dans le tableau 4.1.

Tableau 4.2: Les informations de toutes les instances. [Shuihua Wang et al., 2016]

Instance	Nbr dépôt	Nbr client	Nbr véhicule	Capacité
P01	4	50	4	80
P02	4	50	2	160
P03	5	75	3	140
P04	2	100	8	100
P05	2	100	5	200
P06	3	100	6	100
P07	4	100	4	100
P08	2	249	14	500
P09	3	249	12	500
P10	4	249	8	500
P11	5	249	6	500
P12	2	80	5	60
P13	2	80	5	60
P14	2	80	5	60
P15	4	160	5	60
P16	4	160	5	60
P17	4	160	5	60
P18	6	240	5	60
P19	6	240	5	60
P20	6	240	5	60
P21	9	360	5	60
P22	9	360	5	60
P23	9	360	5	60
Pr01	4	48	1	200
Pr02	4	96	2	195
Pr03	4	144	3	190
Pr04	4	192	4	185
Pr05	4	240	5	180
Pr06	4	288	6	175
Pr07	6	72	1	200
Pr08	6	144	2	190
Pr09	6	216	3	180
Pr10	6	288	4	170

La figure 4.5 représente un exemple d'un fichier de test avec 50 clients, où chaque client est présenté sous forme d'un fichier qui comprend : l'identifiant de chaque client, ces coordonnées, leur fenêtre de temps, leur temps de service, les quantités à transporter, la capacité du transport, le type de véhicule, les contraintes de précédence et de succession.

Ces données sont définit comme suit [Web 9] :

La première ligne contient les informations suivantes :

- Type (T) = 2(MDVRP)
- m : nombre de véhicules
- n : nombre de client
- t : nombre de dépôt

Les lignes suivantes contiennent les informations de chaque véhicule :

- D : durée maximale d'un itinéraire.
- Q : charge maximale d'un véhicule.

Les lignes suivantes contiennent pour chaque client, les informations suivantes :

- i : numéro de client.
- X : coordonnées x
- Y : coordonnées y
- d : durée de service
- q : demande
- f : fréquence de visite
- a : nombre de combinaison de visite
- list : liste de toutes les combinaisons de visites possibles
- e : début de la fenêtre horaire
- l : fenêtre horaire de fin

*p01 - Bloc-notes

Fichier Edition Format Affichage Aide

```
T  m  n  t
2  4  50  4
D  Q
0  80
0  80
0  80
0  80
i  X  Y  d  q  f  a  list  e  l
1  37  52  0  7  1  4  1  2  4  8
2  49  49  0  30  1  4  1  2  4  8
3  52  64  0  16  1  4  1  2  4  8
4  20  26  0  9  1  4  1  2  4  8
5  40  30  0  21  1  4  1  2  4  8
6  21  47  0  15  1  4  1  2  4  8
```

Figure 4.5 : Exemple d'instance p01 avec 50 clients

4.4.2 Choix des paramètres

Le réglage des paramètres affectent les performances d'un algorithme.

La méthode de CRO requiert l'ajustement de 8 paramètres, Les deux paramètres α et β contrôlent l'intensification et la diversification.

Le paramètre CollRate est nécessaire pour faire l'équilibrage entre les collisions. Si elle est inférieure à un nombre aléatoire, r, une collision unimoléculaire est choisie. Sinon, une collision intermoléculaire aura lieu.

Les différents paramètres de l'algorithme CRO-MDVRP donnant les meilleurs résultats trouvés sont présentés dans le tableau 4.2. Il faut noter que les valeurs des paramètres CRO-MDVRP dépendent du problème. Les résultats trouvés sont basés sur 10 exécutions indépendantes pour la meilleure combinaison de ces paramètres.

Tableau 4.3 : Les paramètres de CRO-MDVRP

Paramètres	Description	Valeur
FELimit	Nombre d'itération	10000
La taille de la population (Popsize)	Est le nombre initial de solutions générés aléatoirement dans l'espace des solutions	40
CollRate	Le paramètre responsable de choix des réactions unimoléculaires ou intermoléculaires. De plus, il varie entre 0 et 1.	0.01
Initial energy buffer	L'énergie initiale de buffer	0
KELossRate	Est le taux de perte de KE pendant la réaction.	0.01
InitKE	Est la valeur initiale attribuée à chaque élément de KE dans la phase d'initialisation	50000
Decthres	Où α, le seuil de décomposition	6000
Synthres	Où β, le seuil de synthèse	10000

4.4.3 Test et Résultats

Dans cette section, nous allons présenter les résultats obtenus sur les 33 instances. Pour prouver l'efficacité de notre algorithme, nous comparons nos résultats avec différents algorithmes qui sont utilisés pour résoudre MDVRP ainsi que, les meilleure solutions trouvées (BKS : Best Known Solution) dans la littérature.

Pour des raisons de clarté, nous avons divisé les résultats obtenus en fonction du nombre d'instances. Le premier tableau (4.3) contenant les instances de p01 jusqu'à p23 et le tableau (4.4) comprend les instances de pr01 à pr10.

Le tableau 4.3 illustre une comparaison entre les résultats obtenus, où la première colonne représente le nom de l'instance, la deuxième colonne montre les meilleures solutions trouvées (BKS) dans la littérature et les colonnes de trois à sept contiennent les solutions obtenues par notre algorithme CRO-MDVRP et les différents algorithmes utilisés dans la comparaison: MHDT [Shi et al., 2020], GA [Ombuki-Berman et al., 2009], PSO [Shuihua Wang et al., 2016] et GTS [Escobar Veasquez, 2013].

Tableau 4.4: Comparaison des résultats des tests de p01 à p23

Instance	BKS	GA	PSO	MHDT	GTS	CRO-MDVRP
P01	576.87	622.18	576.87	602.05	576.87	**640**
P02	473.53	480.04	473.53	503.26	473.53	**530**
P03	641.19	706.88	641.19	672.37	641.19	**720**
P04	1001.59	1024.78	1001.59	1065.28	1001.04	**1096**
P05	750.03	785.15	750.03	786.25	750.03	970
P06	876.5	908.88	876.5	912.36	856.5	1020
P07	885.8	918.05	885.8	920.24	884.66	**960**
P08	4437.68	4690.18	4437.68	4537.73	4371.66	5150
P09	3900.68	4240.08	3900.22	4065.49	3880.85	4600
P10	3663.02	3984.78	3686.24	3868.75	3629.6	4359
P11	3554.18	3880.65	3554.18	3756.28	3545.18	4250
P12	1318.95	1318.95	1318.95	1389.65	1318.95	**1419**
P13	1318.95	1318.95	1318.95	1375.29	1318.95	**1418**
P14	1360.12	1365.69	1360.12	1423.96	1360.12	**1405**
P15	2505.42	2579.25	2514.97	2638.19	2505.42	**2610**
P16	2572.23	2587.87	2572.23	2689.63	2572.23	**2625**
P17	2709.09	2731.37	2719.77	2837.26	2709.09	2800
P18	3702.85	3903.85	3733.58	3905.32	3702.85	3998
P19	3827.06	3900.61	3827.06	3989.24	3827.06	**4010**
P20	4058.07	4097.06	4153.13	4268.47	4058.07	**4170**
P21	5474.84	5926.49	5902.53	5837.28	5474.84	**5647**
P22	5702.16	5913.59	6163.11	5996.25	5702.16	**5987**
P23	6095.46	6145.58	6719.36	6356.65	6095.46	8047

A partir de tableau 4.4, nous observons que les résultats obtenus par notre approche CRO-MDVRP sont meilleurs que ceux obtenus par l'algorithme MHDT dans plusieurs instances, on trouve aussi que nos résultats sont très proches à l'algorithme GA dans toutes les instances. Donc il n'y a pas de différence significative entre nos résultats (CRO-MDVRP), BKS et les autres algorithmes de comparaison.

Pour rendre nos résultats faciles à comprendre, nous avons présenté les résultats obtenus graphiquement, Pour clarifier la différence entre les algorithmes. Comme le montre la figure 4.6

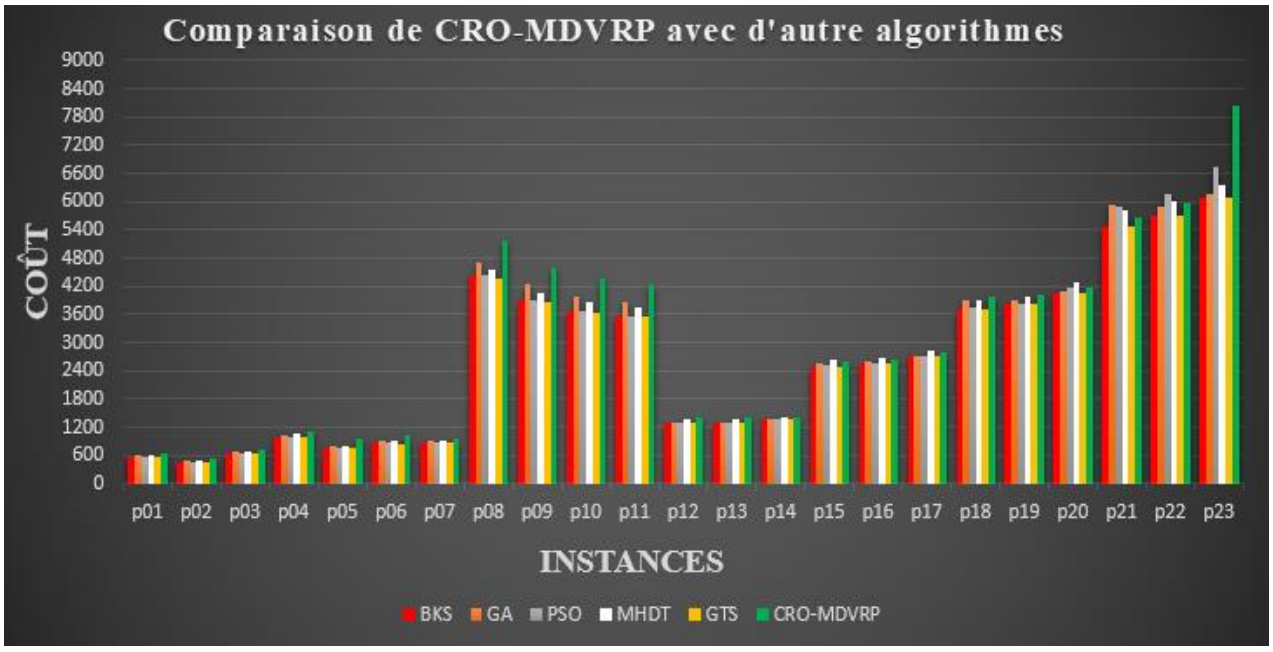

Figure 4.6 : Comparaison des algorithmes pour les tests p01 à p23

La figure 4.6 montre une courbe qui représente la différence entre les résultats de notre algorithme CRO par rapport à BKS et les autres algorithmes en termes de coût au cours des instances. On constate qu'à partir de l'instance p01 à p07 et p12 à p22 les résultats sont plus proches aux BSF. Ensuite, nous avons voir une petite différence entre les résultats de p08 à p11 et p23.

D'après la figure (4.6), il est clairement que notre approche donne des résultats meilleurs ou proches aux autres algorithmes de comparaison, ce qui reflète l'efficacité de l'algorithme proposé.

Tableau 4.5: Comparaison des résultats des instances de pr01 à pr10

Instance	BKS	GA	PSO	GTS	CRO-MDVRP
Pr01	861.32	861.32	861.32	861.32	**896**
Pr02	1307.61	1307.61	1307.61	1311.11	**1335**
Pr03	1806.6	1806.6	1806.6	1803.8	2012
Pr04	2072.52	2072.52	2072.52	2064.11	2137
Pr05	2385.77	2425.55	2385.77	2349.63	**2499**
Pr06	2723.27	2747.71	2768.14	2710.3	2954
Pr07	1089.56	1094.19	1089.56	1089.56	**1175**
Pr08	1666.6	1666.6	1666.6	1665.5	1821
Pr09	2153.1	2214.52	2232.65	2151.45	2340
Pr10	2921.85	3079	3076.58	2910.78	3195

D'âpres ce tableau, nous conclurons que notre algorithme est capable de trouver des résultats proches de BSF et comparables aux autres algorithmes GA, PS0, GTS dans toutes les instances de test.

La figure 4.7 montre une courbe qui représente les résultats. En effet, statistiquement, cette représentation confirme qu'il n'y a pas une différence significative entre tous les algorithmes.

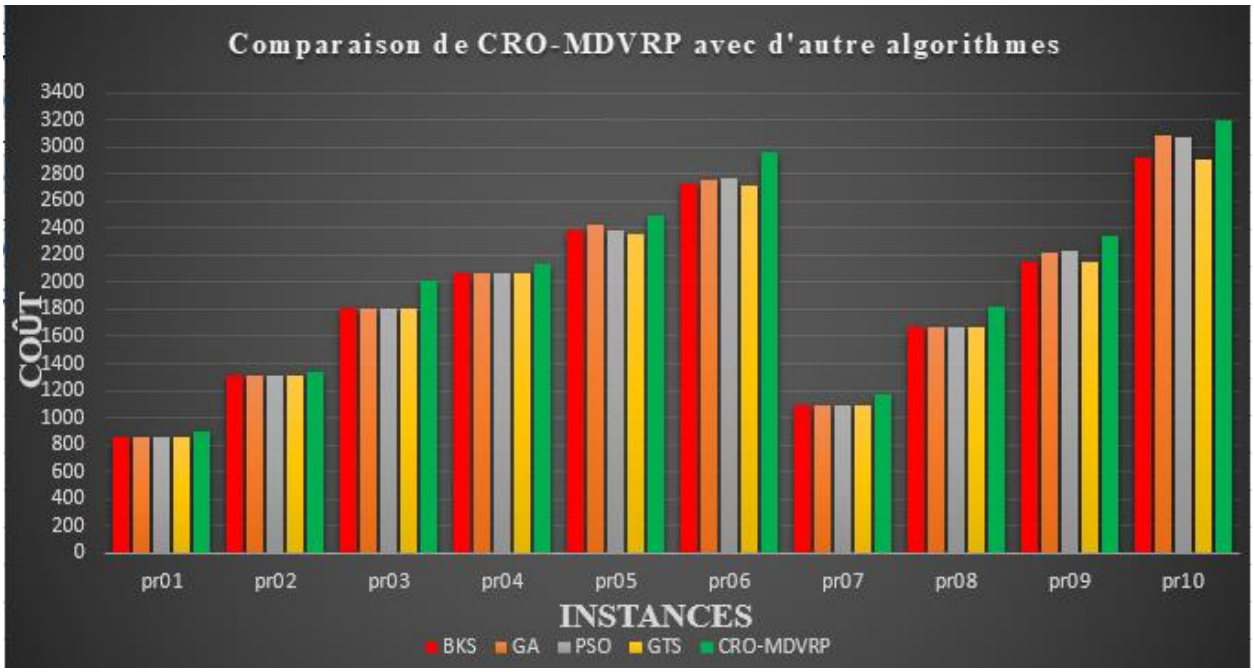

Figure 4.7 : Comparaison des algorithmes pour les tests pr01 à pr10

Globalement, nous pouvons conclure que notre approche est efficace pour résoudre le MDVRP, elle a des performances comparables à celle des autres approches qui sont conçues pour résoudre ce type de problème.

4.5 Conclusion

Dans ce chapitre nous avons présenté l'environnement de développement ainsi que le langage de programmation utilisé. Ensuite, nous avons fait une étude expérimentale et nous avons discuté les résultats obtenues. Finalement, nous avons présenté notre application et expliqué en quelques lignes le rôle de chaque interface.

L'étude faite avec des autres algorithmes de la littérature montre l'efficacité de l'algorithme CRO pour résoudre MDVRP.

Conclusion générale et perspectives

Dans ce mémoire, nous nous sommes intéressés à la résolution d'un problème de la logistique de transport. L'intérêt aux problèmes de la logistique du transport est grandissant dans nos jours. De nombreux domaines sont impliqués à savoir la distribution et la collecte de produits.

De manière plus générale, en logistique du transport, le but est de visiter un ensemble de lieux appelés clients à un coût minimum. On trouve plusieurs problèmes dans la logistique de transport parmi eux le problème de tournées de véhicules multi-dépôts (MDVRP) qui consiste à élaborer des tournées de véhicules associées à chaque dépôt afin de satisfaire la demande d'un ensemble de clients. L'objectif est de minimiser un coût total de la livraison.

Notre travail a pour objectif le développement d'une méthode pour résoudre le problème de tournées de véhicules multi dépôts, ceci nous a amené de découvrir une nouvelle approche et à enrichir notre savoir et notre expérience.

Après une présentation de l'état de l'art sur problèmes d'optimisation, et plus particulière les problèmes d'optimisation combinatoire, on a mis l'accent sur les principales approches utilisées pour la résolution de ces problèmes et on a distingué deux catégories d'approches : les méthodes exactes et les méthodes approchées.

Le problème étudié dans ce travail appartient à la classe des problèmes dire NP-difficiles d'où il n'existe pas des algorithmes qui donnent la solution optimale en un temps raisonnables. Cela nous conduit à utiliser des méthodes approchées dans le but de trouver une solution proche de l'optimale dans un temps raisonnable. Pour cela, Nous avons basé sur la méta-heuristique inspirée des réactions chimiques CRO et nous avons expliqué les différentes phases de développement de cet algorithme.

Cet algorithme s'inspire des réactions chimiques et la distribution de l'énergie entre les molécules. CRO est un algorithme récent qui prouve son efficacité de résoudre des différents problèmes d'optimisation combinatoire. Nous avons adapté les différents opérateurs chimiques selon le contexte du problème étudié.

A la lumière de l'étude expérimentale faite, dans les 33 instances testées, les résultats obtenus comparés à ceux des différentes approches proposées dans la littérature sont très satisfaisants et encourageants.

Les résultats obtenus traduisent l'importance de CRO pour être considéré comme un outil servant à la résolution des autres problèmes d'optimisation combinatoire rencontrés dans différentes domaines.

A l'issue de notre travail, les principales difficultés rencontrées peuvent être résumées comme suit :

- Vu qu'il n'y a pas assez de temps, il n'a pas été possible d'améliorer bien les résultats à travers des différentes méthodes de recherche locale.
- Le nombre de paramètres élevés de CRO qui rend son utilisation dans la résolution de problèmes difficile. Cela est l'inconvénient majeur à améliorer dans cet algorithme

Enfin, dans une perspective d'amélioration de l'approche proposé, nous envisageons de :

- L'amélioration des performances de l'algorithme proposé par l'intégration de méthodes de création de solutions initiales de bonnes qualités.
- L'amélioration des différents opérateurs de CRO pour obtenir des meilleurs résultats.
- Tester notre approche sur des cas réels de transport de marchandises.

Bibliographie

[Alinagian et Shokouhi, 2018] Alinagian, M. a. (2018). Multi-depot Multi-compartement Vehicule Routing Problem, Solved by A hybrid Adaptaive Large Neighborhood Search. *Omega, 76*, 85-99.

[Baldacci et al., 2011] Baldacci R., M. A. (2011). An exact method for the capacitated location-routing problem. *Operations Research, 59*(5), 1284-1296.

[Baldacci et Migozzi, 2009] Baldacci, R. a. (2009). A unified exact methode for solving different classes of vehicule routing problems. *Mathematical Programming, 120*(2), 347-380.

[Barma et al., 2019] Barma, P. D. (2019). A 2-opt guided discrete antlion optimization algorithm for multi-depot vehicule routing problem. . *Dicision Making: Applications in Management and Engoneering, 2*(2), 112-125. Récupéré sur https://doi.org/10.31181/dmame1902089

[Bellman, 1958] Bellman, R. (1958). Dynamic programming and Stochastic control processes. *Information and Control, 1*(3), 228-239.

[Bezerra et al., 2018] Bezerra, S. S. (2018). A GVNS Algorithm for Solving the Multi-depot Vehicule Routing Problem. *Electronic Notes in Discrete Mathematics, 66*, 167-174.

[Brando, 2006] Brandao J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research, 173*, 540–555.

[Carpaneto et Toth, 1980] Carpaneto, G. a. (1980). Some new branching and bounding criteria for the asymmetric traveling salseman problem. *Management Science, 26*, 736-743.

[Chao et al., 1996] Chao, I. M. (1996). The team orienteering problem. *European Journal of Research, 88*(3), 646-674.

[Chao et al., 1993] Chao, I.-M. B. (1993). A new heuristic for the multi-depot vehicule routing problem that improves upon best-known solutions. *American Jornal of Mathematical and Management Scineces, 13*, 371è406.

[Christofides et Eilon, 1969] Christofides, N. e. (1969). An algorithm for the Vehicule-dispatching problem. *Operational Research Quarterly, 20*, 309-318.

[Contrado et Martinelli, 2014] Contrado, C. a. (2014). A new Exact Algorithm for The Multi-depot Vehicule Routing Problem Under Capacity and route Length Constraints. *Discrète Optimization, 12*, 129-146.

[Cordeau et al., 1997] Cordeau, J. M. (1997). A tabu search heuristic for periodic and multi-depot vehicule routing problems. *Networks, 30*, 105-119.

[Crispin et Brando, 2005] Crispim J. and J. Brandao. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the OR Society, 56*(11), 1296–1302.

[Dantiz et Ramser, 1959] Dantiz, G. a. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80-91.

[Devarenne, 2007**]** Devarenne I. Études en recherche locale adaptative pour l'optimisation combinatoire, *Thèse de doctorat*, 30 novembre 2007.

[Dhanens et al., 2002] Dhaenens C., E. M. (2002). Problèmes combinatoires classiques. In Recherche opérationnelle et réseaux : méthodes d'analyse spatiale. *Hermès Science Publications*.

[Alisa et Alison, 1960] Doig, A. L. (1960). Une méthode automatique de résolution des problèmes de programmation discrète. *Econométrie, 28*(3), 497-520. doi:10.2307/1910129

[Dorigi, 1992] Dorigo, M. A. (1992). Distributed optimization by Ant Colonies. Proceedings of the 1rst European Conference on Artificial Life. *Proceedings of the 1rst European Conference on Artificial Life.*, (pp. 134-132). Elsevier Publishing.

[Dorigi et al., 1996] Dorigo, M. V. (1996). Ant System: Optimization by a colony of coperating agents. *IEEE Transactions On Systems, Man, and Cybernetics, part B (Cybernetics), 26*(1), 29-41.

[Dueck, 1993] Dueck, G. (1993). New optimization heuristics: The greet algorithm and the record-to-record travel. *Journal of Computational Physics, 104(1)*, 86-92.

[Escobar, 2013] Escobar Veasquez, J. W. (2013). Heuristic algorithms for the Capacited Location-Routing Problem and the Multi-depot Vehicule Routing Problem.

[Fonseca et Fleming, 1993] Fonseca C. M et Fleming P. J.(1993). Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion and Generalization. In Proceedings of the Fifth Internationl Conference on Genetic Algorithms. pp. 416-423. San Mateo, California.

[Feigenbaun et Feldman, 1963] Feigenbaum, E. A. (1963). (Edirors). Computers and Though. McGraw-HillINC. 6, New York.

[Francis et Slitowitz, 2006] Francis, P. K. (2006). The period vehicule Routing problem with service choice. *Transportation Science, 40*(4), 439-454.

[Glover, 1986] Fred, G. (1986). Future paths for integer programming and links to artificial intelligence. *Computer Operations Research, 13*, 553-549.

[Garey et Johnson, 1979] R. M. Garey et D. S. Johnson. (1979). Computers and Intractability. A guide to the Theory of NP-Completness. W.H.Freeman and CO, San Francisco.

[Gillett et Johnson, 1976] Gillett, B. .. (1976). Multi-terminal vehicule-dispatch algorithm. *Omega, 4*, 711-718.

[Goldberg, 1989] Goldberg, D. F. (1989). Genetic algorithm in search, optimization, and machine learning. Addision Wesley, reading, MA, USA.

[Golden et al., 1977] Golden, B. L. (1977). Implementing Vehicule Routing Algorithms. *Networks, 7*, 113-148.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial System, Ann Arbor: The University of Michigan Press.*

[Kabachi et al., 2017] Kaabachi, I. J. (2017). An Improved Ant Colony Optimization for Green Multi Depot Vehicle Routing Problem with Time Windows. . *IEEE Journal*, 26-28.

[Kennedy et Russell, 1995] Kennedy, J. a. (1995). Particle swarm optimisation. *Proceedings of the IEEE International Conference Neural Networks, 4*, pp. 1942-1995.

[Kilby et al., 1998] Kilby, P. P. (1998). Dynamic VRPs: A Study of Scenarios.

[Kripatrick et al., 1983] Kirkpatrick, S. G. (1983). Optimization by simulated annealing. *Jornal of Science, 220*(4598), 671-680.

[Lacomme et al., 2005] Lacomme, P. C.-C. (2005). competitive memtic algorithms for arc routing problems. *Annals of Operational Research, 165*(2), 535-553.

[Lam et Li, 2012] Lam, A. a. (2012). Chemical reaction optimization: a tutorial. *Memtric Comp, 4*, 3-17. doi:10.1007/s12293-012-0075-1.

[Lam et Li, 2012a] Lam Albert, J. J. (2012). Readed-Coded Chemical Reaction Optimization. *IEEE Transactions on Evolutionary Computation, 16*(3), 339-353.

[Lam et Li, 2010] Lam, A.-Y. a.-V. (2010). Chemical-reaction-inspired metaheuristic for optimization. *IEEETransactions on Evolutionary Computation, 14*(3), 381-399.

[Laporte, 2009] Laporte, G. (2009). Fifty years of vehicle routing. Transportation Science. *Transportation Science, 43*(4), 408-416.

[Laporte et al., 1984] Laporte, G. Y. (1984). Optimal Solutions to Capacited Multi-depot Vehicule Routing Problems. *"Congressus Numerantium, 44*, 283-292.

[Laporte et al., 1988] Laporte, G. Y. (1988). Solving a Family of Multi-depot Vehicule routing and Location-routing Problems. *Transportation Science, 22*, 16-172.

[Lenstra et kan, 1981] Lenstra, J.K., & Kan, A.H. (1981). “Complexity of vehicle routing and scheduling problems“. *Networks, 11*, 221-227.

[J.Li et al., 2015] Li, j. P. (2015). Iterated local serach embdded adaptive neighborhood selection approch for the multi-depot vehicule routing problem with simultaneous deliveries and pickups. *Expert Sustems with Applications, 42*(7), 3551-3561. Récupéré sur https://doi.org/10.2016/j.eswa.2014.12.004.

[Hoff et Lokketangen, 2006] Løkketangen, H. A. (2006). Creating lasso-solutions for the traveling salesman problem with pickup and delivery by tabu searc. *Central European Journal of Operations Research, 14*, 125-140.

[Mirabi et al., 2010] M. Mirabi, S. F. (2010). Efficient Stochastic hybrid heuristics for the multi-depot vehicule routing problem. *Robotics and Computer-Integrated Manufacturing, 26*, 564-569.

[Mathieu, 2008] Mathieu, B. (2008). Méthode de comparaison statique des performances d'algorithms évolusionnaires. Thèse de Doctorat. Université du Québec. Canada.

[Mingozii, 2005] Mingozzi, A. (2005). The multi-depot periodic vehicule routing problem. Book Chapter (Abstraction, Reformulation and approximation),. 347-350.

[Montoya-Torres et al., 2015] Montoya-Torres, J. J.-P. (2015). A litterature Review on the Vehicule Routing Problem with Miltiple Depot. *"Computers & Industrial Enginneering, 79*, 115-129.

[Metropolis et al., 1953] N. Metropolis, A. R. (1953). Equation of state calculations by fast computing machines. *Journal of Chem. Physics, 21*(66), 1087-1092.

[Nagy et Salhi, 2005) Nagy.G. and Salhi.S. (2005). "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries, "European Journal of Operational Research 162: 126-141.

[Oliveira et al., 2016] Oliveira, F. B. (2016). A cooperative Coevolutionary algorithm for the multi-depot vehicule routing problem. *Expert System With Applications, 43*, 117-130. Récupéré sur https://doi.org/10.1016/j.eswa.2015.03.030

[Ombuki-Berma et al., 2009] Ombuki-Berman, B. &. (2009). Using genetic algorithms for multi-depot vehicule routing. *In Bio-Inspired algorithmes for the vehicule routing problem*, 77-99.

[Papadimitriou et Steiglitz, 1982] Papadimitriou, C. a. (1982). Combinatorial optimization-algorithms and complexity. *Prentice Hall*.

[Papadimitrou, 1994] Papadimitrou, C. H. (1994). *Computational Complexity*. Addison-Wesley.

[Parragh et al., 2006] Parragh, S. K. (2006). A survey on pickup and delivery models Part I: Transportation between custumers and depots.

[Raft, 1982] Raft O. M. (1982). A modular algorithm for an extended vehicule scheduling problem. *European Jornal of OperationalResearch, 11*, 67-76.

[Ralphs, 2003] Ralphs Ted, L. W. (2003). On the Capacited Vehicule Routing Problem. *Mathematical Programming, 94*(2), 343-359. doi:10.1007/s10107-002-0323-0

[Renaud et al., 1996] Renaud, J. G. (1996). A tabu search heuritic for the multi-depot vehicule routing problem. *Computers & Operations Research, 23*, 229-235.

[Russell, 1977] Russell R.A. (1977). An effective heuristic for the m-tour traveling salesman problem with some slide conditions. *Operations Research , 25*, 517-524.

[Lin, 1965] S, L. (1965). Computer solutions of the traveling salesman problem. *Beel System Technical Journal, 44*, 2245-2269.

[Saifullah et Islam, 2016] Saifullah, C. K. (2016). Chemical reaction optimization for solving shortest commonsupersequence problem. . *Computational Biology and Chemistry, 64*, 82-93.

[Salhi et Sari, 1997] Salhi, S. a. (1997). A multi-level Composite heuristic for the multi-depot vehicule fleet mix problem. *European journal of Operational Research, 103*, 95-112.

[Sakarovitch, 1984] M. Sakarovitch. (1984). Optimisation combinatoire, Méthodes mathématiques et algorithmiques - Programmation discrète. Hermann. ISBN: 2-7056-5976-5.

[Sombunthan et Kachitvichyanukul, 2010] Sombuntham, P. a. (2010). Sombuntham, P. and Kachitvichyanukul, V. Multi-depot vehicle routing problem with pickup and delivery requests', . *IAENG Transactions on Engineering Technologies, 5*, 71-85.

[Sun et al, 2011] Sun, J. W. (2011). Hybrid algorithm based on chemical reaction optimization and Lin-Kernighan local search for the traveling salesman problem . *in Seventh International Conference on Naturel Computation*, 1518-1521.

[Talbi, 2009] Talbi, E.-G. (2009). Metaheuristics: From Design to Implementation, Wiley. *11*.

[Tellman, 1969] Tellman, F. (1969). The multiple terminal delivrey problem with probabilistic demands. *Transportation Science, 3*, 192-204.

[Tellman et Heiring, 1971] Tellman, F. A. (1971). A study of look-ahead procedure for solving the multiterminal delivey problem. *Transportation Research, 5*, 225-229.

[Tellman et Cain, 1972] Tellman, F. e. (1972). An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science, 18*, 664-682.

[Toth et al., 2002] Toth, P. a. (2002). An overview of vehicule routing problem. In The vehicule rounting problem. *Society for Industrial and Applied Mathematics*, 1-26.

[Toth et Vigo, 2003] P. Toth & D. Vigo. (2003). The Granular Tabu Search and it application to the vehicule-routing problem. *INFORMS Journal on Computing*, 15(4), 333-346.

[Throng et al., 2013] Truong, T.-K. L. (2013). Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem. . *Applied Soft Computing, 13*(4), 1774-1780.

[Tu et al., 2014] Tu, W. F.-L. (2014). A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicule routing problem. *Transportation Research part E: Logistics and Transportation Review, 61*, 84-97. Récupéré sur https://doi.org/10.1016/j.tre.2013.11.003

[Cerny, 1985] V.Cerny. (1985). Thermodynamical Approch to the traveling salseman problem: An efficient simulation algorithm. *Jornal of Optimization Theory and Applications, 45*(1), 41-51.

[Veeramachaneni et Osadan, 2005] Veeramachaneni, K. L. (2005). “An Adaptive Multimodal Biometric Management Algorithm,”. *IEEE Trans. Sys. Man & Cyber., 35*(3), 344-356.

[Shuihua Wang et al., 2016] Wang, S. L. (2016). Fitness-scaling adaptive genetic algorithm with local search for solving the Multiple Depot Vehicle Routing Problem. *Simulation, 92*(7), 601-616.

[Wang et al., 2016] Wang, X. G. (2016). The min-max split delivery milti-depot vehicule routing problem with minimum service time requirement. *Computers & Operations Research*, *71*, 110-126. Récupéré sur https://doi.org/10.1016/j.cor.2016.01.008

[Wren et Holiday, 1972] Wren, A. e. (1972). Computer scheduling of vehicules from one or more depots to a mumber of delivery points. *23*, 333-344.

[Xioa et al., 2003] Xiang Xiao, E. R.-M. (2003). Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. *IEEE, In Proceeding International Paralle and Distributed Processing Symposium*, 10. doi:10.1109/IPDPS.2003.121390

[Xin, 2010] Xin-SheYang. (2010). Engineering Optimization: An Introduction with Metaheuristic Applications - Published by John Wiley & Sons, Inc., Hoboken, New Jersey,ISBN 978-0-470-58246-6.

[XU et al., 2013] Xu, Y. L.-T. (2013). A dag scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization. *J. Parallel Distrib. Comput. , 73*(9), 1306-1322.

[Shi et al., 2020] Yanjun Shi, L. L. (2020). A Heuristic Solution Method for Multi-Depot Vehicule Routing-Based Waste Collection Problems. *Applied Sciences, 10*(7), 2403.

Webographie

[Web1] Optimisation et Recherche Operationnelle. (2022, fevreir 4). Récupéré sur Introduction a l'optimisation: https://www.youtube.com/watch?v=qKgZRQMBpxQ&list=PL74OzRodm1vB99SriXfSbPrNYHYgo8cnS&index=2&t=2s

[Web2] https://fr-academic.com/dic.nsf/frwiki/376914

[Web3] https://fr.wikipedia.org/wiki/Probl%C3%A8me_P_%E2%89%9F_NP

[Web4] https://favpng.com/png_view/classical-centralforce-problem-vehicle-routing-problem-ant-colony-optimization-algorithms-travelling-salesman-problem-png/QLV30wj9

[Web 5] DM-Pro. (2021, Avril 30). Récupéré sur Algorithme de colonie de fourmis pour le problème TSP| intelligence artificial|Application java: https://www.youtube.com/watch?v=ZlSLVYDupm8&t=272s

[Web 6] Orkia Derkaoui. (2021, Aout 15). Récupéré sur La Méthode de descente et La Recherche Locale: Pourquoi Recherche Locale, Notion de voisinage 2/8.: https://www.youtube.com/watch?v=R15dHkKOznY

[Web 7] Orkia Derkaoui. (2021, Aout 13). Récupéré sur La Méthode de descente et La Recherche Locale: Pourquoi Recherche Locale, Notion de voisinage 1/8: https://www.youtube.com/watch?v=k4qnsG9L_Gc&t=420s

[Web 8] Orlia Derkaoui. (2021, mai 10). Récupéré sur Méthodes d'Optimisation Méthode de Recherche Tabou Principe de l'Algorithme Liste des Tabou Partie 3: https://www.youtube.com/watch?v=A61LMBP4br4&t=408s

[Web 9] VRP Web (dorronsoro.es)