N° d'ordre : 03/2015-D/INF

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Électronique et de l'Informatique

THESE

Présentée pour l'obtention du **grade** de **DOCTEUR EN SCIENCE**

EN : INFORMATIQUE
Spécialité : Systèmes Informatiques

Par : Berghida Meryem

Sujet :

Stratégies adaptatives et coopératives pour la résolution de problèmes de tournées de véhicules avec collecte et livraison

Soutenue publiquement, le 25/04/2015, devant le jury composé de :

MelleBoughaci Dalila	Professeur	à USTHB	Présidente
MBoukra Abdelmadjid	Professeur	à USTHB	Directeur de thèse
M Baba Ali Riad	Professeur	à USTHB	Examinateur
M Ait Zai Hakim	Maitre de Conférence/A	à USTHB	Examinateur
MBatouche Mohamed Chawki	Professeur	à l'univ de Constantine	Examinateur
Mme Kamel Nadjet	Maitre de Conférence/A	à l'univ de Sétif	Examinatrice

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse, Prof. BOUKRA Abdelmadjid, pour son encadrement, sa disponibilité, son soutien et ses conseils tout au long de ces trois années de recherche. Il m'a apporté une compréhension plus approfondie des différents aspects de la recherche.
Merci à Madame Prof. Boughaci Dalila d'avoir présidé le jury. Merci également aux messieurs Prof. BATOUCHE Mohamed Chawki, Dr. AIT ZAI Hakim, Prof. BABA ALI Riadh et madame Dr.KAMEL Nadjet. pour avoir accepté de participer au jury.

Je remercie également les membres du laboratoire des systèmes informatique dont j'ai fait parti durant trois années de recherche, mes collègues de la salle des doctorants Asma Bellili, Hadjer Lacheheb, Aicha Boutorh, Wassila Guendouzi et Meriem Khalifa pour leur encouragements et leur aide.

J'adresse aussi ma gratitude la plus sincère à mes parents, pour leur aide, leur bienveillance et surtout leur confiance et leur amour. Mes remerciements à mes soeurs Imène, Nour El Houda, Amira et mon petit frère Ahmed Yasser pour leur présence et leur aide précieuse.

Mes derniers remerciements et non les moindres, s'adressent à mon oncle maternel Djamel et sa famille, pour son accueil, son soutien, ses encouragements et son aide. je lui serai éternellement reconnaissante d'avoir été présent pour moi durant ces années de recherche.

Liste des tableaux

Table des figures

Liste des Algorithmes

Table des matières

Introduction générale

Les problèmes logistiques abondent dans de nombreux domaines tels que les problèmes de planification, les problèmes de routage, les problèmes d'ordonnancement...etc. ; mais les bonnes solutions à ces problèmes ne sont pas si nombreuses, et ont à la fois un intérêt commercial et scientifique. Le sous-ensemble de problèmes de routage a reçu une attention particulière, en commençant par le fameux problème du voyageur de commerce pour englober de plus en plus une gamme plus large de problèmes de tournées de véhicules.
Le transport de marchandises et de passagers est une tâche importante dans la société d'aujourd'hui. Des sommes astronomiques sont dépensées quotidiennement sur le matériel, son entretien et les salaires. Il est donc évident de tenter de réduire la somme d'argent dépensée pour le transport, aussi, même de petites améliorations peuvent conduire à d'énormes économies. Plusieurs approches pourraient être adoptées. On pourrait améliorer le matériel ou améliorer les infrastructures. On pourrait aussi envisager d'améliorer l'organisation en adoptant des techniques de recherche opérationnelle (RO). Compte tenu des ressources limitées disponibles, il serait intéressant d'envisager la troisième issue. Toth et Vigo [Toth 2001] estiment que l'utilisation des procédés informatiques pour la planification du processus de distribution conduit souvent à des économies dans le domaine de 5% à 20% des frais de transport, de sorte que l'étude de ces procédures est très intéressante.

Dans de nombreux problèmes de décision, comme dans le domaine de gestion de la production et de la logistique, l'évaluation des alternatives et la détermination d'une solution optimale ou au moins quasi-optimale est une tâche aussi importante que difficile.

Pour la plupart de ces problèmes, aucun algorithme efficace pour les grandes instances n'est connu, et les approches classiques de la recherche opérationnelle comme la programmation linéaire mixte en nombre entier ou la programmation dynamique sont souvent d'une utilité limitée en raison du temps de calcul excessif. Par conséquent, des approches heuristiques de résolution ont été développées. Elles visent à fournir de bonnes solutions dans un délai raisonnable pour un problème donné.

Toutefois, ces méthodes ont deux inconvénients majeurs : Le premier inconvénient est qu'elles sont adaptées à un problème spécifique, et que leur adaptation à d'autres problèmes est difficile et souvent même impossible. Le deuxième inconvénient est qu'elles sont généralement conçues pour construire une solution unique dans la manière la plus efficace, alors que la plupart des problèmes de décision ont un grand nombre de solutions faisables. Il y a donc de fortes chances que d'autres meilleures solutions existent. Pour résoudre ce

problème, des stratégies de recherche indépendantes du problème, en particulier, les métaheuristiques ont été proposées.

Nous nous intéressons dans cette thèse à la résolution d'un des problèmes de logistique, à savoir le problème de tournées de véhicule avec collecte et livraison. Ce problème est caractérisé par un ensemble de véhicules desservant deux types de clients (clients de collecte et clients de livraison). Ces véhicules ont un dépôt commun. La livraison des différents clients se fait au moyen de tournées. Le problème consiste à minimiser le coût de ces tournées. Ce problème étant classé NP-difficile [Laporte 1992], nous avons opté pour l'utilisation des métaheuristiques dans sa résolution.

L'objectif de cette thèse est de prospecter la voie de la coopération et de l'adaptation dans la résolution du problème de tournées de véhicules (VRP). Pour cela, nous avons résolu deux variantes du VRP, à savoir, le problème de tournées de véhicules avec flotte hétérogène, collecte et livraison et fenêtres de temps (HVRPMBTW) et le problème de tournées de véhicules avec collecte et livraison simultanées (VRPSPD). Les deux problèmes ont en commun l'aspect collecte et livraison. Le problème HVRPMBTW consiste à servir deux types de clients avec une flotte de véhicules limitée et hétérogènes, où chaque client est servi dans un intervalle de temps précis sans contrainte de précédence. Le problème VRPSPD consiste de servir un ensemble de clients avec une flotte homogène et illimitée de véhicules. Chaque client étant un point de collecte et de livraison. Le premier problème a été abordé à l'aide de métaheuristiques relativement récentes, en utilisant le concept de l'adaptation dans le but d'évaluer le comportement de ces nouvelles métaheuristiques pour ce problème. Le deuxième problème a été traité en utilisant la stratégie de coopération entre les métaheuristiques.

Le concept de l'adaptation consiste à faire évoluer dynamiquement le comportement de la métaheuristique en fonction de son environnement. Dans notre cas, nous avons opté pour le réglage dynamique des paramètres de la métaheuristique en fonction de l'instance utilisée. Le concept de coopération consiste à faire coopérer plusieurs métaheuristiques esclaves dont le fonctionnement est supervisé par une métaheuristique maitre dans le but de profiter des avantages offerts par chacune des métaheuristiques esclaves.

Cette thèse est organisée comme suit : Dans le premier chapitre, nous décrivons le problème de tournées de véhicule VRP avec ses principales variantes ainsi que quelques méthodes de résolution présentes dans la littérature. Notre contribution est présentée dans le deuxième et le troisième chapitre. Dans le chapitre 2, nous développons l'approche adaptative proposée pour résoudre le problème HVRPMBTW, pour cela, nous présentons les quatre approches proposées, à savoir, l'approche biogéographie améliorée, l'approche recherche coucous discrète, l'approche discrète de recherche d'harmonie améliorée et

enfin l'approche recherche d'harmonie quantique avec population de taille variable. En fin de ce chapitre, nous présentons une étude comparative entre ces quatre approches ainsi qu'une étude statistique. Dans le chapitre 3, nous présentons l'approche coopérative proposée pour la résolution de VRPSPD en utilisant quatre types de métaheuristiques. Nous concluons cette thèse dans le chapitre 4.

Chapitre 1

Problème de Tournées de Véhicules

Sommaire

Les dernières décennies ont vu une utilisation croissante des méthodes d'optimisation, basées sur la recherche opérationnelle pour la bonne gestion de fourniture des biens et services dans les systèmes de distribution. L'utilisation des procédures informatisées pour la planification de distribution génère des économies consistantes dans les coûts de transport mondial, ce dernier représente un élément pertinent (généralement de 10% à 20%) du coût final des marchandises. Dans cette section, nous nous focalisons sur les problèmes de distribution des biens entre des centres de dépôts et les clients. Ces problèmes sont connus par les problèmes de tournées de véhicules. La version originale de ce problème a été proposée par Dantzig et Ramser [Dantzig 1959]. Après la publication de leur article, qui avait l'objet de calculer un ensemble de routes optimales pour une flotte de camions de livraison, le VRP a attiré l'attention

d'un grand nombre de chercheurs vue sa ressemblance aux problèmes rencontrés dans le monde réel par les entreprises de distribution et de transport.

1.1 Le problème de tournées de véhicules

Le problème de distribution des biens dans lequel les véhicules basés à un dépôt central sont tenus de visiter (pendant une période de temps) des clients géographiquement dispersés afin de satisfaire les exigences des clients connus est appelé le VRP (Vehicle Routing Problem). Le problème de tournées de véhicules (VRP) est l'un des plus célèbres problèmes d'optimisation combinatoire, étant considéré comme une extension de TSP, il est donc un problème **NP-Difficile**. Ainsi, on peut le décrire brièvement comme suit :
Compte tenu d'un ou de plusieurs centres de dépôts, une flotte de véhicules (homogènes ou non), et un ensemble de clients ayant des besoins connus ou prévus ; l'objectif est de déterminer un ensemble de routes fermées (se terminant à l'un des centres de dépôts) avec un coût global minimal qui peut répondre aux demandes des clients dispersés géographiquement, tout en satisfaisant les contraintes de capacité des véhicules et du dépôt. D'autres exigences peuvent être aussi considérées, comme : la restriction du temps de voyage, l'ordre pour visiter les clients, la définition des fenêtres temporelles en spécifiant la période souhaitable de parvenir à un endroit donné...etc., ce qui donne un ensemble riche de variantes de VRP.

1.2 Variantes du problème de tournées de véhicules

Le problème de base de tournées de véhicules enchaine un certain nombre d'hypothèses, comme l'utilisation d'une flotte homogène, un seul dépôt, une route par véhicule...etc. Ces hypothèses peuvent être enrichies en introduisant des contraintes supplémentaires au problème comme :

- La taille de la flotte disponible : un seul véhicule, plusieurs véhicules
- La composition de la flotte disponible : homogène (un seul type de véhicule), hétérogène (plusieurs types de véhicules).
- Le nombre des centres de dépôt : dépôt unique, plusieurs dépôts.
- La nature de demande : déterministe ou aléatoire.
- La distribution : sur les sommets, le long des arcs, mixte.
- La nature de réseau : dirigé, non dirigé, mixte euclidien.
- La capacité des véhicules : imposées (identique ou variante avec le type de véhicule), non imposées

- La durée maximale d'une tournée : imposée et identique pour tous les véhicules, imposée pour chaque type de véhicule, non imposée.
- Le type de service : collecte, livraison, mixte.
- Coût d'une tournée : variable, constant, limité, illimité.
- Temps pour servir un sommet : temps spécifié et fixé à l'avance, non spécifié, un intervalle de temps.
- Objectifs : minimiser le coût total de distribution en termes de distance ou de temps de voyage, minimiser le nombre de véhicules utilisés...etc.

Ceci implique l'augmentation de la complexité du problème, et par conséquent classer les problèmes étendus comme des problèmes NP-difficile. Dans la section suivante, nous allons introduire ces variantes avec leurs formulations mathématiques.

1.2.1 CVRP (Capacitated Vehicle Routing Problem)

Le problème CVRP (Capacitated Vehicle Routing Problem) a d'abord été formulé par [Christofides 1976]. CVRP est la version de base du VRP où tous les clients correspondent à des livraisons. Les demandes sont déterminées et connues à l'avance et ne peuvent pas être divisées. Tous les véhicules sont identiques et possèdent un dépôt unique, où seulement la restriction de la limite de capacité de véhicule est imposée. L'objectif alors est de minimiser le coût total pour servir tous les clients [Toth 2001].
Dans ce qui suit, nous présentons la modélisation mathématique de CVRP proposé par Rego et Roucairol dans [Rego 1994] :
Soit $G = (N, A)$ un graphe où $N = \{1, ..., n\}$ est un ensemble de sommets avec le sommet 1 fixé comme dépôt et $A = \{(i, j)|(i, j) \in N^2 \text{ et } i \neq j\}$ est l'ensemble des arcs. $N' = N \setminus \{1\}$ est l'ensemble des clients à satisfaire obtenu par le fait d'enlever le dépôt de l'ensemble des sommets.
A chaque arc est associé un coût non négatif c_{ij} qui peut représenter le coût du voyage ou la durée du voyage entre i et j. Nous envisageons le cas où une flotte de m véhicules ayant la même capacité de transport D (véhicules homogènes) est disponible.
Les contraintes prises en compte dans cette formulation sont :

- **La contrainte de capacité :** A chaque sommet i de N' est associé un poids d_i non négatif représentant la demande. La somme des poids d'une tournée ne doit pas dépasser la capacité D du véhicule.
- **La contrainte de temps total :** Le temps total d'une tournée ne doit pas dépasser une borne T. Ce temps est la somme des temps des voyages entre les sommets et les temps d'arrêt à ces derniers.

La formulation de [Rego 1994] adopte les notations suivantes :
Les constantes de données :

n = nombre de sommets,
m = nombre de véhicules,
D_k = capacité du véhicule k,
T_k = temps maximal de la tournée du véhicule k,
d_i = demande du sommet $i, (d_1 = 0)$,
t_i^k = temps nécessaire au véhicule k pour charger ou décharger au sommet i,
t_{ij}^k = temps nécessaire au véhicule k pour voyager du sommet i au sommet j,
c_{ij} = coût ou distance du voyage du sommet i au sommet j.
N = ensemble des sommets.
X = matrice d'éléments $x_{ij} = \sum_{k=1}^{m} x_{ij}^k$
Les variables de décision :

$$x_{ij}^k = \begin{cases} 1 & \text{si le véhicule } k \text{ voyage du sommet } i \text{ au sommet} j \\ 0 & \text{sinon} \end{cases} \tag{1.1}$$

Avec $X^k = (x_{ij}^k)$.
La fonction à optimiser :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i^j \sum_{k=1}^{n} x_{ij}^k \tag{1.2}$$

Sous :

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ij}^k = 1, j = 2, ..., n \tag{1.3}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{ij}^k = 1, i = 2, ..., n \tag{1.4}$$

$$\sum_{i=1}^{n} x_{ip}^k - \sum_{j=1}^{n} x_{pj}^k = 0, k = 1, ..., m; p = 1, ..., n \tag{1.5}$$

$$\sum_{i=1}^{n} d_i (\sum_{j=1}^{n} x_{ij}^k) \leqslant D_k, k = 1, ..., m \tag{1.6}$$

$$\sum_{i=1}^{n} t_i^k \sum_{j=1}^{n} x_{ij}^k + \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij} x_{ij}^k \leqslant T_k, k = 1, ..., m \tag{1.7}$$

$$\sum_{j=2}^{n} x_{1j}^{k} \leq 1, k = 1, ..., m \tag{1.8}$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1, k = 1, ..., m \tag{1.9}$$

$$X^{k} \in S \tag{1.10}$$

Où S est donné par l'expression (1.11) [Ralphs 2003] :

$$S = \left((x_{ij})^{k} | \sum_{i \in Q} \sum_{j \notin Q} x_{ij}^{k} \geq 1, \forall Q \subset N, |Q| \geq 2; k = 1, ..., m \right) \tag{1.11}$$

Où Q est un ensemble de sommets visités par un seul véhicule.

La fonction objectif (1.2) consiste à minimiser le coût total de transport. Les équations (1.3) et (1.4) assurent que chaque sommet ne soit servi qu'une seule fois par un et un seul véhicule. L'équation (1.5) assure la continuité d'une tournée par un véhicule : le sommet visité doit impérativement être quitté l'équation (1.6) assure le respect de la contrainte de capacité du véhicule. L'équation (1.7) assure le respect de la contrainte de la durée totale d'une tournée. Les équations (1.8) et (1.9) assurent le non dépassement de la disponibilité d'un véhicule. Un véhicule ne sort du dépôt et n'y revient qu'une seule fois. Finalement les équations (1.10), (1.11) assurent le respect des contraintes d'élimination des sous tournées (tournées revenant au client et pas au dépôt). D'une façon générale, le problème CVRP consiste à affecter chaque client à une tournée effectuée par un seul véhicule de capacité fixe. Ce véhicule commence et termine sa tournée au dépôt [Ralphs 2003].

1.2.2 VRPTW (Vehicle Routing Problem with Time Windows)

Le problème de tournées de véhicules avec contrainte de fenêtres de visite est une généralisation de VRP ou bien un prolongement de CVRP. Dans un VRPTW, de nouvelles contraintes temporelles sont ajoutées : chaque client doit être servi dans un intervalle de temps durant lequel il est disponible pour être visité [Cordeau 2001].

Une fenêtre de visite est un intervalle de temps $[a_i, b_i]$ associé à un client i, cet intervalle représente le temps pendant lequel une visite chez lui est possible.

Il existe deux types de fenêtre de visite, la fenêtre de visite large / serrée. D'un côté, la fenêtre de visite large autorise un véhicule à arriver en dehors de la fenêtre de visite des clients mais en contrepartie une pénalité dépendant de la violation sera alors infligée. De l'autre côté, la fenêtre de visite serrée n'autorise en aucun cas l'arrivée d'un véhicule en dehors des heures de visite [Grellier 2008].
Pour étendre le modèle proposé par Fisher et Jaikumar [Fisher 1981] afin de prendre en considération les fenêtres de visite des différents sommets, introduisons les termes suivants :

- a_i : la borne inférieure de la fenêtre de visite du sommet i ;
- b_i : la borne supérieure de la fenêtre de visite du sommet i ;
- s_i : le temps de service du sommet i ;
- t_{ij} : le temps de transport entre les noeuds i et j ;
- u_i^k : l'instant de visite du sommet i par le véhicule k ;
- T : une grande valeur ($T \gg 0$).

Les contraintes de respect des fenêtres de visite peuvent ainsi s'écrire :

$$\forall i \in (1, ..., n), \forall k \in (1, ..., M) a_i \leqslant u_i^k \leqslant b_i \tag{1.12}$$

$$\forall i \in (1...n), \forall j \in (1...n), \forall k \in (1...M) u_i^k + s_i + t_{ij} - T(1 - x_{ij}^k) \leqslant u_j^k \tag{1.13}$$

La contrainte (1.12) permet de vérifier que l'instant de visite d'un client se trouve entre les bornes inférieure et supérieure de la fenêtre de visite du client concerné. La contrainte (1.13) vérifie la cohérence des instants de visite lorsque deux sites se suivent.

1.2.3 VRPB (Vehicle Routing Problem with Backhauls)

Le VRP avec Backhauls VRPB est le prolongement du CVRP, dans lequel l'ensemble des clients $V \setminus \{0\}$ est partitionné en deux sous-ensembles. Le premier sous-ensemble L, contient n clients à servir (Linehauls), chacun nécessitant une quantité donnée de produits à livrer. Le deuxième sous-ensemble B, contient m clients Backhauls, où une quantité donnée de produits entrants doivent être ramassés [Toth 2001]. Les clients sont numérotées de sorte que : $L = 1, ..., n$ et $B = n + 1, ..., n + m$.
Dans le VRPB, une contrainte de précédence entre les clients de collecte (Backhauls) et de livraison (Linehauls) existe : lorsqu'une route contient les deux types de clients, tous les clients de livraison doivent être servis avant tout client de collecte qui peut être servi. Une demande non négative d_i, pour être livrée ou collectée en fonction de son type, est associé à chaque client i, et

le dépôt est associé à une demande fictive $d_0 = 0$. Lorsque la matrice des coûts est asymétrique, le problème est appelé Asymmetric VRP with Backhauls (AVRPB).
Le VRPB (et AVRPB ainsi) consiste à trouver exactement une collection de K circuits simples à un coût minimal, telle que [Toth 2001] :

1. Chaque circuit visite le sommet de dépôt ;
2. Chaque sommet de client est visité exactement par un circuit ;
3. La demande totale des deux types de clients visités par un circuit ne doit pas dépasser séparément, la capacité C du véhicule ;
4. Dans chaque circuit, tous les clients Linehauls (de livraison) précédera la clientèle de collecte, le cas échéant. Les circuits contenant seulement les clients Backhauls généralement ne sont pas autorisés.

Par ailleurs, on observe que la contrainte de précédence (4) introduit une orientation implicite vers des itinéraires de véhicules mixtes, c.-à-d., les routes qui visitent à la fois les sommets de livraison et de collecte.

1.2.4 VRPPD (Vehicle Routing Problem with Pickup and Delivery)

Dans la version de base de VRPPD, chaque client i est associé avec deux quantités d_i et p_i, représentant la demande des produits homogènes à livrer et à collecter au client i, respectivement. Dans certains cas, seulement une quantité de demande $d'_i = d_i - p_i$ est utilisée pour chaque client i, indiquant la différence nette entre les demandes de livraison et de collecte (éventuellement négative). Pour chaque client i, O_i dénote le sommet qui est l'origine de la demande de livraison, et D_i dénote le sommet qui est la destination de la demande de collecte.

On suppose que, à chaque emplacement d'un client, la livraison est exécutée avant la collecte ; donc le chargement courant du véhicule avant son arrivée à un emplacement donné est défini par la charge initiale moins toutes les demandes déjà livrer plus toutes les demandes déjà collectées.
Le VRPPD consiste à trouver exactement une collection de K circuits simples avec un coût minimal tel que :

1. Chaque circuit visite le sommet de dépôt.
2. Chaque sommet de client est visité exactement par un circuit.
3. La charge courante du véhicule tout au long de circuit ne doit pas être négative et ne doit jamais dépasser la capacité C du véhicule.

4. Pour chaque client i, le client O_i doit être servi dans le même circuit et avant le client i ; et
5. Pour chaque client i, le client D_i doit être servi dans le même circuit et après le client i.

Le VRPPD est une généralisation du VRP classique, qui appartient aussi à une large famille de problèmes de livraison et Collecte (PDPs). On peut être distingué trois fameux types de problèmes de collecte et livraison qui ont été étudié dans la littérature. Un est Problème de livraison et collecte à un seul produit (single-commodity PDP) dans lequel un seul type de biens est soit collecté ou bien livré à chaque noeud. C'est le cas par exemple quand une véhicule blindée transporte de l'argent entre des branches de bureaux d'une banque. Une autre variante est le problème de collecte et de livraison à deux produits (two-commodity PDP) où deux types de biens sont considérés et chaque noeud peut agir comme un noeud de livraison et de collecte au même temps. Ce problème survient dans la livraison des boissons quand les véhicules livrent les bouteilles pleines et collectent celles qui sont vides. Une variante de ce problème est le VRP with Backhauls où toutes les livraisons doivent être exécutées avant toute collecte. Finalement, le PDP à n-produits se produit quand chaque produit est associé à un seul noeud de collecte et un seul noeud de livraison. C'est le cas où des voyageurs ou des biens doivent être transportés d'une origine à une destination. D'habitude, ce problème se ramène au le VRPPD.
Puisque la plupart des applications pratiques de VRPPD incluent des restrictions sur le temps dans lequel chaque emplacement peut être visité par un véhicule, il est pratique de présenter une variante générale du problème, nommée VRPPD avec fenêtres de temps (VRPPDTW).

Notons par n le nombre de requêtes à satisfaire. Supposons que tous les véhicules ont un seul sommet de dépôt. Le VRPPDTW peut être défini par un graphe direct $G = (N, A)$ où $N = P \cup D \cup \{0, 2n+1\}, P = \{1, ..., n\}$ et $D = \{n+1, ..., 2n\}$. Les sous ensembles P et D contiennent les noeuds de collecte et de livraison, respectivement, où les noeuds 0 et $2n+1$ présentent le dépôt d'origine et de destination. Avec chaque requête i est ainsi associé un noeud d'origine i et un noeud de destination $n+i$. Supposons K est l'ensemble des véhicules et $m = |K|$. À chaque véhicule $k \in K$ est associée une capacité Q_k et sa durée totale à ne pas dépasser en route T_k. Avec chaque noeud $i \in N$ est associé une charge q_i et une durée de service non négative d_i tel que $q_0 = q_{2n+1} = 0, q_i = -q_{n+i} (i = 1, ..., n)$ et $d_0 = d_{2n+1} = 0$. Une fenêtre de temps $[e_i, l_i]$ est aussi associée avec chaque noeud $i \in N$ où e_i et l_i représentent le temps le plus tôt et le plus tard, respectivement, dans lequel

le service peut démarrer au noeud i. A chaque arc $(i,j) \in A$ est associé un coût de routage c_{ij} et un temps de voyage t_{ij}.
Pour chaque arc $(i,j) \in A$ et chaque véhicule $k \in K, x_{ij}^k = 1$ si et si seulement si le véhicule k voyage du noeud i jusqu'au noeud j. Pour chaque noeud $i \in N$ et chaque véhicule $k \in K$. Soit B_{ik} le temps dans lequel un véhicule k commence le service au noeud i, et Q_{ik} est la charge du véhicule k après qu'il visite le noeud i. Le VRPPDTW peut être formulé comme suit [Desaulniers 2001] :

$$min \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij}^k x_{ij}^k \tag{1.14}$$

Sous :

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1, (i \in P) \tag{1.15}$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0, (i \in P, k \in K) \tag{1.16}$$

$$\sum_{j \in N} x_{0j}^k = 1, (k \in K) \tag{1.17}$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0, (i \in P \cup D, k \in K) \tag{1.18}$$

$$\sum_{i \in N} x_{i,2n+1}^k = 1, (k \in K) \tag{1.19}$$

$$B_j^k \geq (B_i^k + d_i + t_{ij}) x_{ij}^k, (i \in N, j \in N, k \in K) \tag{1.20}$$

$$Q_j^k \geq (Q_i^k + q_j) x_{ij}^k, (i \in N, j \in N, k \in K) \tag{1.21}$$

$$B_i^k + d_i + t_{i,n+i} \leq B_{n+i}^k, (i \in P, k \in K) \tag{1.22}$$

$$B_{2n+1}^k - B_0^k \leq (T_k, (k \in K) \tag{1.23}$$

$$e_i \leq B_i^k \leq l_i, (i \in N, k \in K) \tag{1.24}$$

$$max\{0, q_i\} \leq Q_i^k \leq min\{Q_k, Q_k + q_i\}, (i \in N, k \in K) \tag{1.25}$$

$$x_{ij}^k \in \{0, 1\}, (i \in N, j \in N, k \in K) \tag{1.26}$$

La fonction objectif minimise le coût total de routage. Les contraintes (1.14) et (1.15) assurent que chaque requête est servie exactement une fois et que les noeuds de livraison et de collecte associés sont visités par le même véhicule. Les contraintes (1.16)-(1.19) garantissent que la route de chaque véhicule k commence à l'origine dépôt et termine au dépôt destination. La cohérente des variables de temps et de charge est assurée par les contraintes (1.20) et (1.21). La contrainte (1.22) force les véhicules à visiter le noeud de collecte d'une demande avant son noeud de livraison. Enfin, l'inégalité (1.23) limite la durée de chaque route alors que (1.24) et (1.25) impose la fenêtre de temps et les contraintes de capacité, respectivement.

1.2.5 OVRP (Open Vehicle Routing Problem)

Le problème de tournées de véhicule ouvert (OVRP) est une variante du problème de routage de véhicule classique où les véhicules ne reviennent pas au dépôt après le service des clients. L'application d'OVRP dans la vie réelle s'agit du cas où soit l'entreprise n'a pas du tout de véhicules, ou les véhicules appartenant à l'entreprise ne sont pas suffisants pour pouvoir les utiliser dans la distribution des produits aux clients. Dans les deux cas, l'entreprise doit louer un certain nombre de véhicules afin de réaliser la distribution des produits. Lorsque les véhicules terminent leur tournée, ils ne reviennent pas au dépôt. Du point de vue de l'optimisation combinatoire, la principale différence entre le CVRP et la OVRP est que dans le premier cas, la route est un cycle hamiltonien tandis que dans le second cas, la route est un chemin hamiltonien [Marinakis 2013].
Dans le problème OVRP, l'objectif est de minimiser le nombre de véhicules et de minimiser la distance totale (ou le temps) voyagé. Chaque itinéraire commence au dépôt et se termine à un client, visitant un certain nombre de clients, chacun une fois, sans retour au dépôt. La demande de chaque client doit être complètement traitée par un seul véhicule. La demande totale desservie par chaque véhicule ne doit pas dépasser la capacité du véhicule. En outre, dans une variante du problème, le temps du voyage de chaque véhicule ne doit pas dépasser une limite maximale [Yu 2011].
La formulation générale est décrite comme suit [Sariklis 2000] : Soit n le

nombre de clients, $N = 1, ..., n$ l'ensemble des clients, et $N_0 = 0, 1, ..., n$ l'ensemble de tous les clients et le dépôt. Ce dernier est identifié par 0. Soit $K = \{1, 2, ..., k\}$ l'ensemble des véhicules. Soit d_i la demande du client $i \in N$, soit l_h la charge du véhicule $h \in K$. En outre, $max(d_i) \leq max(l_h)$. Définissant :

$$y_{ih} = \begin{cases} 1 & \text{si le sommet } i \text{ (client ou dépot)est visité par le véhicule } h \\ 0 & \text{sinon} \end{cases} \tag{1.27}$$

$$x_{ijh} = \begin{cases} 1 & \text{si le véhicule } h \text{ voyage du sommet } i \text{ au sommet } j \\ 0 & \text{sinon} \end{cases} \tag{1.28}$$

Soit c_{ij} le coût de transport (comme la distance, le temps...etc.) entre les clients i et j. Le modèle mathématique du problème OVRP est décrit comme suit :

$$\min Z = \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{h=1}^{k} c_{ij} x_{ijh} \tag{1.29}$$

Sous :

$$\sum_{i=0}^{n} d_i y_{ih} \leq l_h, \forall h \in K \tag{1.30}$$

$$\sum_{h=1}^{k} y_{ih} = 1, i = 1, ..., n \tag{1.31}$$

$$\sum_{i=0}^{n} x_{ijh} = y_{ih}, j = 1, ..., n; \forall h \in K \tag{1.32}$$

$$\sum_{j=1}^{n} x_{ijh} = y_{ih}, i = 1, ..., n; \forall h \in K \tag{1.33}$$

$$x_{ijh} = 0 \text{ or } 1; i, j = 1, ..., n; \forall h \in K \tag{1.34}$$

$$y_{ih} = 0 \text{ or } 1; i, j = 1, ..., n; \forall h \in K \tag{1.35}$$

Dans le modèle, la formule (1.29) est la fonction objectif qui minimise la distance totale traversée. La contrainte (1.30) porte sur la capacité des véhicules. La contrainte (1.31) indique que chaque noeud de demande doit être servi ; (1.32) et (1.33) assurent que chaque noeud de demande est servi par exactement le même véhicule.

1.2.6 DVRP (Dynamic Vehicle Routing Problem)

La première référence à un problème de routage dynamique du véhicule est due à Wilson et Colvin [Wilson 1977]. Ils ont étudié le problème DARP à un seul véhicule, dans lequel les demandes des clients sont des voyages à partir d'une origine vers une destination qui apparait dynamiquement. Leur approche utilise des heuristiques capables d'obtenir de bons résultats avec un temps de calcul raisonnable. Plus tard, Psaraftis [Psaraftis 1980] a introduit le concept de demande immédiate : un client qui demande le service veut toujours être desservi le plus tôt possible, ce qui nécessite une re-planification immédiate de la route actuelle du véhicule.
Le problème de tournées de véhicule dynamique (DVRP) [Psaraftis 1995] est fortement liée à la VRP statique, car il peut être décrit comme un VRP dans lequel des informations sur le problème peuvent changer au cours du processus d'optimisation [Khouadjia 2010].
DVRP est un problème dynamique en temps discret, et peut être considéré comme une série d'instances de P ; chaque cas est un problème statique, qui commence à l'instant t et doit être résolu dans un délai spécifique de Δ_t Nous le résumons comme suit [Psaraftis 1995] :

$$P = \{(P_i, t_i, \Delta_t), i = 0, 1, ..., i_{max}\} \tag{1.36}$$

Avec cette information, la durée de l'instance i est $t_{(i+1)} - t_i$. Le nombre maximum d'instances i_{max} peut-être infini si le problème est ouvert. Une nouvelle instance est générée par l'action de modification d'environnement ρ_i sur l'instance i. Cela se traduit par $P_{(i+1)} = \rho_i \oplus P_i$. Ce changement dans l'environnement peut être dû à plusieurs facteurs, par exemple, les temps de déplacement peuvent prendre du temps [Haghani 2005] ou à cause des problèmes liés au trafic [Rochat 1995], les commandes peuvent être annulées ou modifiées [Wang 2007], certains clients peuvent être inconnus lorsque l'exécution commence [Sun 2007], etc.

1.2.7 HVRP (Heterogeneous Vehicle Routing Problem)

Le HVRP ou bien le problème de tournées de véhicules avec flotte hétérogène, diffère du VRP classique dans le type de flotte de transport. Il traite une flotte hétérogène de véhicules ayant différentes capacités, des coûts fixes et des coûts variables.
Le HVRP implique donc la planification d'un ensemble de tournées de véhicule où chacune débute et se termine au dépôt, pour une flotte hétérogène de véhicule qui servent un ensemble de clients ayant des exigences connues.

Chaque client est visité exactement une fois. La demande totale d'un itinéraire ne doit pas dépasser la capacité du véhicule qui lui était affecté. Le coût de calcul d'itinéraire d'un véhicule est la somme de son coût fixe et un coût variable proportionnel à la distance parcourue. L'objectif est de minimiser la somme de ces coûts de routage. Il existe deux versions du problème dans la littérature : HVRP avec flotte limitée et HVRP avec flotte illimitée.

1.3 Les méthodes de résolution de VRP et ses variantes

1.3.1 Travaux sur l'état de l'art

Dans [Laporte 1992], quelques résultats principaux connus relatifs au VRP sont examinés. Toth et Vigo ont revues les méthodes exactes les plus efficaces dans la littérature jusqu'en 2002 dans un chapitre dans leurs livre [Toth 2001]. Une autre revue sur le problème VRPPD s'est trouvée sur deux parties dans [Parragh 2008]. Ces dernières années ont vu une attention accrue sur le VRP intégré avec des contraintes de chargement supplémentaires, connus sous le nom de 2L-CVRP ou 3L-CVRP. Dans [Wang 2009], un état de l'art sur 2L-CVRP et 3L-CVRP est présenté. Dans [Pillac 2013], les auteurs classent les problèmes de routage de point de vue qualité de l'information et évolution, après avoir présenté une description générale du routage dynamique, ils introduisent la notion de degré de dynamisme et présentent une revue complète d'application et méthodes de résolution des problèmes de tournées de véhicules dynamique. Le document [Baldacci 2012] présente une revue des développements récents qui ont eu un impact majeur sur l'état de l'art des algorithmes exacts actuels résolvant le VRP. Les auteurs passent en revue les formulations mathématiques, relaxations et méthodes exactes récentes de deux des variantes les plus connus de VRP : CVRP et VRPTW. Une revue de littérature sur les développements et les publications récentes portant sur le VRP est effectuée dans [Panneerselvam 2012].
Compte tenu de la sensibilité environnementale des problèmes de tournées de véhicule, une revue approfondie de la littérature des problèmes de tournées de véhicules verts (Green VRP) est présenté dans [Lin 2014].

1.3.2 Les méthodes exactes

En 2008, Baldacci et Mingozzi [Baldacci 2008] ont présenté une méthode exacte unifiée pour résoudre un modèle élargi du problème HVRP (H pour hétérogène) nommé HDVRP qui contient comme cas spéciaux : The Single De-

pot CVRP, SDVRP (Site-Dependant VRP), MDVRP (Multi-Depot VRP). Un modèle de programmation linéaire en nombre entier mixte pour le SVRPPD (Single VRPPD) est proposé dans [Gribkovskaia 2007]. Il introduit la notion de solution générale qui englobe les formes de solution connues telles que : hamiltonien, double trajet...etc. Une variante du VRP appelée VRPTW avec multiples routes est abordé dans [Macedo 2011]. Ils considèrent qu'un véhicule donné peut être attribué à plus d'un itinéraire par période de planification. Ils proposent un nouvel algorithme exact pour ce problème. L'algorithme est itératif et s'appuie sur un modèle de flux de réseau pseudo polynomiale. Dans [Bettinelli 2011], les auteurs présentent l'algorithme de Branch and Cut and Price pour trouver une solution exacte d'une variante de VRPTW, où le flot de transport contient des véhicules avec différentes capacités et des coûts fixes.

1.3.3 Les méthodes approchées

Un problème d'optimisation combinatoire NP-difficile est difficile à résoudre de manière exacte pour les grandes instances. Les méthodes exactes nécessitent un grand temps de calcul qui croit exponentiellement avec la taille des instances du problème. C'est pour cela qu'on fait appel aux méthodes approchées (heuristiques et métaheuristique).
Dans ce qui suit, nous citons les principaux travaux qui traitent le problème du VRP et ses variantes par les méthodes approchées.
En 1999, un algorithme heuristique a été développé [Toth 1999] pour résoudre le problème de VRPB symétrique et asymétrique. Pour la même classe (VRPB), une heuristique unifiée a été proposée [Ropke 2006]. Dans [Nagy 2005], les auteurs proposent une méthode qui trouve une solution au problème de VRP et modifie cette solution pour qu'elle soit possible pour le VRPPD. Dans [Geem 2005], l'algorithme de recherche d'harmonie est appliqué sur le problème VRP. En 2006, un algorithme mémétique [Tavakkoli-Moghaddam 2006] utilisant différents algorithmes de recherche locale est proposé pour résoudre le VRPB.
De nombreuses façons de modélisation des contraintes de collecte ont été proposées dans la littérature, imposant chacune des restrictions différentes concernant le traitement des clients de collecte. Une étude de ces modèles est présentée et un modèle unifié est développé dans [Ropke 2006]. Ce modèle est capable de gérer la plupart des variantes du problème dans la littérature. Un nouvel algorithme évolutionnaire quantique (Quantum Inspired Evolutionary Algorithm QIEA) est présenté dans [Hu 2009] pour le VRPSPD (Vehicle Routing Problem with Simultaneous Pickup and Delivery).
En 2009, un système d'aide à la décision DSS est développé [Tütüncü 2009] afin de résoudre le VRPB, VRPB mixte et le VRPB restreint (le VRPB res-

treint est un problème compromis entre VRPB classique et le VRPB mixte), et un nouveau critère qui considère la capacité restante des véhicules est proposé pour la création des solutions pour le VRPB mixte et restreint.
Un algorithme quantique d'optimisation par essaims particulaires est proposé dans [Zhengchu 2010] pour résoudre le problème de CVRP. De même, pour résoudre la même variante (CVRP), un algorithme de colonies de fourmis modifié est proposé dans [Chen 2012a]. Il consiste à minimiser la distance totale parcourue par chaque véhicule ainsi que le temps total de service sur les noeuds clients. Le CVRP étudié est traité en deux phases : phase d'affectation client/véhicule et phase de minimisation du temps d'exécution.
En 2011, le Framework d'optimisation localisée (LOF) [Ursani 2011] est introduit. Ce Framework est une procédure itérative entre deux phases d'optimisation de De-optimisation. Pour tester leur hypothèse, ils ont choisi un algorithme génétique comme méthodologie d'optimisation et VRPTW comme espace de domaine. Ce nouveau schéma de l'algorithme est appelé LGA (Localized Genetic Algorithm).
Le problème de tournées de véhicule avec collecte et livraison sélectif SPDP est une nouvelle variante de VRPPD. Ce problème relaxe la contrainte que tous les noeuds de collecte doivent être visités le long du chemin. Plus précisément, le SPDP a pour but de trouver le chemin le plus court qui peut servir les noeuds de livraison avec les produits nécessaires à partir de quelques noeuds de collecte sélectionnés. Dans l'étude de [Liao 2012], ils proposent une mutation adaptative qui se concentre sur la sélection de noeuds de collecte appropriés pour le SPDP. Deux algorithmes évolutionnaires (algorithme génétique et algorithme mémétique) sont développés.
Dans l'article de [Naji-Azimi 2013], un programme linéaire basé sur une approche heuristique est proposé. Ce programme peut être utilisé comme un outil complémentaire pour améliorer la performance des méthodes existantes qui résolvent ce problème. En 2012, les auteurs dans [Hong 2012] étudient le problème de tournées de véhicule dynamique avec fenêtres de temps (DVRPTW). Un algorithme amélioré de recherche de voisinage large (ILNS) est proposé. Dans la même année, une nouvelle version modifiée de GRASP nommé (MPNS-GRASP) pour résoudre le CVRP est proposé dans [Marinakis 2012].
En 2013, l'article [Franceschelli 2013] aborde une nouvelle classe des problèmes de routage de véhicules multiples hétérogènes. Les auteurs proposent deux algorithmes distribués basés sur la communication potinée (Gossip communication). Le premier algorithme est basé sur une optimisation locale exacte et le deuxième est basé sur une heuristique gloutonne approximative locale.
Ran Liu et al. [Liu 2013a] traitent dans leur article un problème d'ordonnancement de véhicules rencontré dans la logistique des soins à domicile. Il concerne la fourniture des médicaments et de dispositifs médicaux de la phar-

macie de l'entreprise des soins à domicile aux maisons des patients, livraison des médicaments spéciaux de l'hôpital aux patients et collecte d'échantillons biologiques, des médicaments non utilisés et des dispositifs médicaux de patient. Le problème peut être considéré comme un VRP spécial avec livraison et collecte simultanée et fenêtres de temps avec quatre types de demandes : livraison de dépôt au patient, livraison de l'hôpital à un patient, collecte de patient au dépôt et collecte de patient au laboratoire médicale. Deux modèles de programmation entière mixte sont proposés. Ils proposent ensuite un algorithme génétique et une méthode taboue. Un algorithme (max-min Ant System) pour résoudre le SDWVRP (Split Delivery Weighted Vehicle Routing Problem, qui consiste à construire des routes optimales en respectant les contraintes de capacité de véhicule et le poids de chargement pour servir un ensemble de client à coût minimal) est proposé dans [Tang 2013]. Dans [Cao 2014], les auteurs étudient l'OVRP avec des demandes incertaines, où les véhicules ne reviennent pas nécessairement à leur emplacement d'origine après la livraison des marchandises aux clients. Ils proposent quatre stratégies robustes pour faire face à l'incertitude de la demande et une amélioration de l'algorithme d'évolution différentielle IDE pour résoudre le modèle robuste d'optimisation. Dans [Gulic 2013], la programmation génétique est utilisée pour faire évoluer une heuristique qui crée des solutions initiales pour différents objectifs et différentes classes de VRP.

1.3.4 Les méthodes hybrides

Russell et al. [Bent 2006] proposent un algorithme hybride à deux phases pour résoudre le VRPPD. La première phase utilise un algorithme simple de recuit simulé pour diminuer le nombre des routes, et la deuxième phase utilise la recherche de voisinage large (LNS) pour diminuer le coût du voyage total. Un algorithme évolutionnaire quantique hybride HQIEA (Hybrid Quantum Inspired Evolutionary Algorithm) est proposé dans [Zhang 2008] pour résoudre le CVRP. Un autre algorithme HQIEA (Hybrid Quantum Inspired Evolutionary Algorithm) est proposé dans [Zhang 2009a], avec optimisation de 2-opt pour résoudre l'OVRP. Le HQIEA est utilisé pour l'exploration globale, l'algorithme 2-opt est utilisé pour optimiser les routes afin d'accélérer la convergence. Dans [Subramanian 2013a], un algorithme hybride pour une classe de problème de tournées de véhicules avec flotte homogène est proposé. Une séquence de modèles de Set-Partitioning (SP) avec des colonnes correspondant aux itinéraires trouvés par une approche métaheuristique sont résolus, pas nécessairement à l'optimalité en utilisant le solveur à MIP (Mixed Integer Programming), qui peut interagir avec la métaheuristique lors de son exécution. L'algorithme proposé a été testé sur plusieurs variantes de VRP

(CVRP, OVRP, VRPPD,... etc.). Deux variantes de HVRP sont traitées dans [Liu 2013b]. L'une avec des coûts fixes et des coûts variables (HVRPFD), et l'autre avec seulement des coûts variables. Une heuristique de population hybride qui est capable de résoudre les deux variantes est proposée, où une population de solutions est progressivement améliorée par croisements et recherches locales.

1.4 Conclusion

Dans l'économie moderne, les produits sont achetés dans le monde entier en utilisant le transport et l'organisation du transport a pris de plus en plus d'attention. Le transport ne cause pas seulement des coûts, mais augmente également la circulation, la pollution de l'environnement, le bruit...etc. Une organisation de transport inefficace peut entraîner une diminution du niveau de service. Tous ces précédents facteurs nous incitent à optimiser les tournées de véhicules. L'optimisation des transports est un problème fréquemment étudié dans le contexte de la logistique. Il se pose non seulement dans la collecte et la distribution de marchandises, mais aussi dans les applications de services comme le taxi, les livraisons de colis, le routage de bus scolaires...etc.
Une entreprise qui fait face à des problèmes de transport fait essentiellement face à trois types de limitations différentes : Ressources limitées imposées par la société elle-même (Un certain nombre de conducteurs et de véhicules, La capacité du véhicule (poids, volume), Restrictions de chargement (c'est à dire l'empilement n'est pas autorisé)...etc.). Les exigences imposées par la loi(Le temps de conduite maximale autorisée, Restriction de la conduite sur les routes (limite de poids, ou ils ne doivent pas utilisé à certains moments)...etc.). Enfin, les clients pourraient avoir besoin à être desservis à un certain moment (intervalle), ou avant ou après un certain temps. Il peut aussi y avoir des restrictions de chargement imposées par les clients comme certains types de chargement et/ou des dispositifs de déchargement qui ne sont pas disponibles sur le site du client.

Ce sont des contraintes populaires qui peuvent survenir dans le contexte des problèmes de tournées de véhicules. Bien sûr, on peut aussi faire face à des restrictions et exigences complètement différentes car elles dépendent du problème.
Avant de prendre une décision tous ces aspects doivent être pris en considération. Par conséquent, le processus de planification devient très compliqué, surtout si un calendrier de routage doit être créé à partir de zéro, ou lorsque les demandes disposant d'une forte variation quotidienne de planification manuelle peuvent utiliser la main-d'oeuvre qui peut être utilisée ailleurs.

Dernièrement, le nombre d'applications commerciales informatiques disponibles pour le routage du véhicule est en augmentation tout comme l'utilisation des métaheuristiques dans ces applications. Pour cette raison, un problème typique dans le domaine de la logistique de transport est étudié dans les chapitres suivants.

Chapitre 2

Approche adaptative pour la résolution du problème HVRPMBTW

Dans ce chapitre, nous traitons une variante intéressante, mais rarement traitée du problème de tournées de véhicules nommé problème de tournées de véhicules avec flotte hétérogène, collecte et livraison et fenêtres de temps HVRPMBTW. Le problème étudié est inspiré des travaux antérieurs de Belmecheri et al. [Belmecheri 2013]. Ce problème considère deux types de clients : les clients recevant les produits (linehauls customers), et les clients possédant des produits à collecter au centre de dépôt (backhauls customers). Il est à noter que les produits livrés aux clients de livraison sont différents de ceux qui sont ramassés des clients de collecte. Les biens livrés à partir du dépôt central peuvent être alternés avec les biens collectés. Le service à n'importe quel client commence dans un intervalle de temps donné (par exemple, si un véhicule arrive trop tôt à un client, il faut attendre que la fenêtre de temps s'ouvre, et il n'est pas permis d'arriver en retard). Les véhicules assurant le service ont différentes capacités et des coûts variables.

2.1 Formulation mathématique du premier problème traité

Dans cette section, HVRPMBTW est défini et formulé comme un programme mathématique, ce modèle est pris des travaux de Farah Belmecheri [Belmecheri 2013] :

Data

n	Nombre de clients
$0, n+1$	Noeud de début et d'arrivée
V	Ensemble des noeuds $V = \{0, 1, ..., n, n+1\}$
D_c	Sous ensemble des clients de livraison
P_c	Sous ensemble des clients de collecte

A	Ensemble des arcs $A = \{(i,j) : i,j \in V, i \neq n+1, j \neq 0, i \neq j\}$
a_i	Temps au plutôt pour commencer le service au noeud i
b_i	Temps le plus tard pour commencer le service au noeud i
s_i	Temps de service du client i
d_i	Quantité livrée au client $i \in D_c, d_i = 0$ Si $i \in P_c \cup \{0\} \cup \{n+1\}$
p_i	Quantité collectée chez un client $i \in P_c, p_i = 0$ Si $i \in D_c \cup \{0\} \cup \{n+1\}$
t_{ij}	Temps de voyage du client i vers le client j
c_{ij}	Coût de l'arc (i,j)
K	Ensemble des véhicules utilisés
Q_k	Capacité du véhicule k
z_k	Coût variable par distance d'un véhicule k
M	Constante de grande valeur positive

Variables

$T_i^k \geq 0$	Début du service du véhicule k chez un client i
$L_i^k \geq 0$	Charge du véhicule k quand il quitte le client i
$x_{ij}^k \in \{0,1\}$	$x_{ij}^k = 1$ si et seulement si l'arc (i,j) est traversé par le véhicule k.

Il est à noter que le dépôt est représenté par deux noeuds, 0 pour l'origine des routes et $n+1$ pour leur destination. Nous considérons un graphe complet orienté dans lequel les arcs non utilisés sont supprimés (c.à.d. les boucles, les noeuds entrant au noeud 0 et les noeuds sortant du noeud $n+1$).
Nous supposons que l'arc (i,j) représente le chemin le plus court avec le coût c_{ij} et le temps t_{ij} dans le réseau considéré.
Les graphes non orientés peuvent être traités par la mise de $c_{ij} = c_{ji}$ et $t_{ij} = t_{ji}$ pour chaque arc (i,j). Notez que la distance entre deux noeuds est euclidienne.
Le modèle mathématique est le suivant :

$$min \sum_{k \in K} \sum_{j:(i,j) \in A} c_{ij} x_{ij}^k z^k \tag{2.1}$$

Sous les contraintes suivantes :

$$\sum_{k \in K} \sum_{j:(i,j) \in A} x_{ij}^k = 1 \quad , \forall i \in D_c \cup P_c \tag{2.2}$$

$$\sum_{i:(0,i) \in A} x_{0j}^k = 1 \quad , \forall k \in K \tag{2.3}$$

$$\sum_{i:(0,n+1) \in A} x_{in+1}^k = 1 \quad , \forall k \in K \tag{2.4}$$

$$\sum_{j:(j,i)\in A} x_{ji}^k = \sum_{j:(i,j)\in A} x_{ij}^k \quad ,\forall k \in K, \forall i \in D_c \cup P_c \tag{2.5}$$

$$a_i \leq T_i^k \leq b_i \quad ,\forall k \in K \tag{2.6}$$

$$T_i^k + s_i + t_{ij} \leq T_j^k + M(1 - x_{ij}^k) \quad ,\forall (i,j) \in A \tag{2.7}$$

$$p_i \leq L_i^k \leq Q_k \quad ,\forall k \in K \tag{2.8}$$

$$0 \leq L_i^k \leq Q_k - d_i \quad ,\forall i \in D_c, \forall k \in K \tag{2.9}$$

$$L_i^k + p_i - d_j \leq L_j^k + M(1 - x_{ij}^k) \quad ,\forall (i,j) \in A, \forall k \in K \tag{2.10}$$

$$L_0^k = \sum_{i\in D_c} d_i \sum_{j:(i,j)\in A} x_{ij}^k \quad ,\forall k \in K \tag{2.11}$$

$$x_{ij}^k \in \{0,1\} \quad ,\forall (i,j) \in A, \forall k \in K \tag{2.12}$$

La fonction objectif (2.1) représente la distance totale pondérée par le coût du véhicule utilisé. La contrainte (2.2) assure que chaque client est visité par un et seulement un véhicule. Les contraintes (2.3, 2.4) assurent que chaque véhicule commence à partir du noeud 0 et termine au noeud $N+1$. La contrainte (2.5) assure que si un véhicule k arrive chez un client i, il doit le quitter. La contrainte (2.6) respecte la fenêtre de temps, tandis que la contrainte (2.7) assure la propagation de temps de passage. Les deux contraintes suivantes (2.8, 2.9) vérifient la capacité du véhicule après la visite d'un client. La contrainte (2.10) assure la propagation de la charge du véhicule tout au long du chemin. La charge de chaque véhicule quand il quitte le dépôt doit être égale à la demande totale des clients de livraison. Ceci est assuré par la contrainte (2.11). Enfin, les variables binaires sont déclarés dans la dernière contrainte.

2.2 Technique d'adaptation

Un système adaptatif s'adapte aux conditions qui changent l'environnement en changeant sa structure. En effet, chaque système se prépare à des changements en utilisant un plan d'adaptation (c.à.d l'ensemble des facteurs qui contrôlent ce changement) [Holland 1975]. Le plan d'adaptation

détermine la façon dont les structures sont modifiées pour mieux s'adapter aux changements d'environnement. En règle générale, les plans d'adaptation sont réalisés en élaborant des opérateurs déterminant la façon avec laquelle les changements de structures seront réalisés.
L'adaptation dans les algorithmes inspirés de la nature peut prendre différentes formes. Par exemple, les manières d'équilibrer l'exploration et l'exploitation constituent la forme fondamentale de l'adaptation [MITRE 2005]. Comme la diversité peut être intrinsèquement liée à l'adaptation, il est préférable de ne pas examiner ces deux caractéristiques séparément. Si l'exploitation est forte, le processus de recherche va utiliser les informations de problème spécifique obtenues au cours du processus itératif pour guider les nouveaux mouvements de recherche, cela peut conduire à une recherche focalisée et donc de réduire la diversité de la population, qui peut aider à accélérer la convergence de la procédure de recherche. Toutefois, si l'exploitation est trop forte, il peut en résulter la perte rapide de la diversité de la population et peut donc conduire à la convergence prématurée. D'autre part, si les nouveaux mouvements de recherche ne sont pas guidés par des informations de paysage local, il se peut, généralement, que la capacité d'exploration soit augmentée et que de nouvelles solutions avec une grande diversité soient générées. Cependant, trop de diversité et d'exploration peuvent aboutir à des chemins de recherche inutiles, ainsi conduire à une convergence lente. Par conséquent, l'adaptation de mouvements de recherche afin d'équilibrer l'exploration et l'exploitation est important. Donc, maintenir la diversité équilibrée dans une population est également important.
Cependant, comment parvenir à un tel équilibre est encore un problème ouvert. En fait, aucun algorithme ne peut prétendre avoir atteint un tel équilibre optimal dans la littérature actuelle. Essentiellement, l'équilibre est lui-même un problème d'hyper-optimisation, car il est l'optimisation d'un algorithme d'optimisation. En outre, un tel équilibre peut dépendre de nombreux facteurs tels que le mécanisme de travail d'un algorithme, le réglage et le contrôle de ses paramètres et même le problème considéré. En outre, un tel équilibre ne peut exister universellement [Wolpert 1997], et il peut varier d'un problème à un autre, nécessitant ainsi une stratégie d'adaptation.

Dans la littérature, plusieurs travaux ont utilisé des stratégies adaptatives pour résoudre des problèmes d'optimisation. Dans [Brest 2015], les auteurs donnent un aperçu des mécanismes adaptatifs et auto-adaptatifs dans les algorithmes d'évolution différentielle DE. Ces travaux montrent que ces méthodes se basent principalement sur un contrôle des paramètres de mutation et de croisement, mais peu sur la taille de la population. En outre, le papier propose un nouvel algorithme jDE auto-adaptatif en utilisant deux

stratégies, sélectionnés par la même probabilité durant l'exécution. Dans [Fister Jr. 2015], les auteurs s'intéressent à l'auto-adaptation des paramètres de contrôle dans les stratégies d'évolution (ES). Dans cette optique, une analyse des opérateurs de mutation est effectuée. Le papier [Trunfio 2015] présente une revue des techniques adaptatives les plus pertinents tel que proposé dans les papiers abordant la co-évolution coopérative (CC) dans les algorithmes évolutionnaires. La CC divise la population en sous-populations qui explorent l'espace de recherche dans des directions différentes. Ces sous-populations coopèrent en échangeant des informations afin d'évaluer la qualité des individus. Le papier présente ainsi un nouvel algorithme CC adaptatif de FA (Firefly algorithm).

Il est bien connu que le choix des paramètres des métaheuristiques a un impact direct sur les performances de recherche, et cela a conduit à un grand intérêt au divers mécanismes, qui, d'une certaine manière tentent de régler automatiquement les paramètres de l'algorithme. Bien sûr cela a fait apparaître le spectre du paramétrage inapproprié résultant d'un mauvais choix de la technique d'adaptation.
Les valeurs appropriées des paramètres changent en changeant l'instance du problème, c'est pour cela qu'on a choisi d'adapter les valeurs des paramètres avec l'instance du problème en utilisant l'approche de descente (Hill Climbing). Chaque solution S est un vecteur de paramètres de l'approche utilisée. Nous intégrons l'algorithme de réglage des paramètres au début de chaque métaheuristique. Le choix de la descente est motivé par son absence de paramètres. L'algorithme 1 résume le réglage des paramètres par la descente.

Algorithme 1 Algorithme de réglage des paramètres (la descente)d'une métaheuristique M avec n paramètres

Générer une solution initiale aléatoirement $S = [Parametre_1^*, ..., Parametre_n^*]$
Appliquer l'algorithme M et calculer F^*
Générer K solutions voisines de la solution S tel que une solution voisine $S_k = [Parametre_{1,k}, ..., Parametre_{n,k}]$
Pour chaque solution voisine k :
Appliquer l'algorithme M et calculer son F_k^*
si $F_k^* > F^*, k \in K$ **alors**
$Parametre_1^* = Parametre_{1,k}^*, ..., Parametre_n^* = Parametre_{n,k}^*, F^* = F_k^*$
Répéter jusqu'à il n'y aura plus d'amélioration dans F^*

2.3 Première approche de résolution : Algorithme d'optimisation amélioré basé sur la biogéographie EBBO

2.3.1 BBO : Biogeography Based Optimization (Optimisation Basée sur la Biogéographie)

BBO est un nouvel algorithme inspiré de la biogéographie pour l'optimisation globale. La biogéographie étudie la répartition géographique d'organismes biologiques. il est dû aux travaux d'Alfred Wallace [Wallace 1876a], [Wallace 1876b] et Charles Darwin [Darwin 1995] dans le 19 ème siècle. Ces travaux ont eu un aspect descriptif et un aspect historique. En 1960, Robert MacArthur et Edward Wilson ont développé un modèle mathématique pour la biogéographie [MacArthur 1967].
Les modèles mathématiques de la biogéographie décrivent comment les espèces migrent d'une île à l'autre, comment de nouvelles espèces apparaissent et comment les espèces s'éteignent [Simon 2008].
Une île est un habitat, un habitat est un espace de vie isolé des autres espaces. Les habitats qui sont favorables à la résidence des espèces biologiques ont un HSI (Habitat Suitability Index) élevé. Les paramètres influençant le HSI peuvent être : la pluie, la diversité végétale, la diversité des terrains ... etc. Les variables décrivant l'habitabilité sont appelées SIV (Suitability Index Variables). Les habitats sont caractérisés par ce qui suit :

- Un habitat avec un HSI élevé tend à avoir un nombre élevé d'espèces, alors que celui avec un HSI bas tend à avoir moins d'espèces.
- Les habitats avec un HSI élevé sont caractérisés par un taux élevé d'émigration et un taux bas d'immigration parce qu'ils sont saturés.
- Les habitats avec un HSI bas ont un taux élevé d'immigration et un taux bas d'émigration. L'immigration doit entraîner la modification du HSI de l'habitat.Les espèces qui vivent dans un habitat qui reste trop longtemps sans amélioration ont tendance à disparaître.

Dan Simon a introduit en 2008 une métaheuristique basée sur la biogéographie. Il utilise l'analogie suivante :

- Une solution est analogue à un habitat.
- La qualité (fitness) d'une solution est analogue au HSI.
- Les variables définissant la solution sont analogues au SIVs.
- Une bonne solution est analogue à un habitat avec un HSI élevé, un nombre élevé d'espèces, un haut taux d'émigration et un taux bas d'immigration.

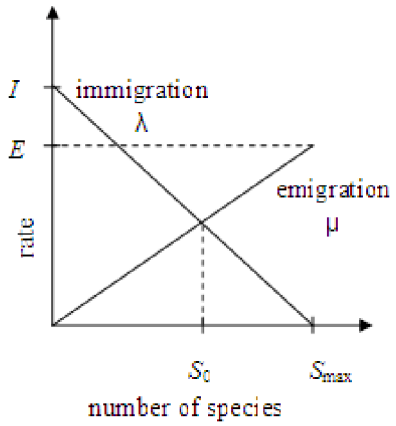

FIGURE 2.1 – Évolution des taux d'émigration et immigration en fonction du nombre d'espèces

- Une mauvaise solution est analogue à un habitat avec un HSI bas, un nombre bas d'espèces, un taux bas d'émigration et un taux élevé d'immigration.
- Une bonne solution tend à partager ses caractéristiques avec une mauvaise solution pour l'améliorer (migration des SIVs). Ceci est analogue à la migration des espèces entre les habitats. Partager les caractéristiques n'entraine pas un changement dans les caractéristiques de la bonne solution, puisque la migration se fait en utilisant seulement un échantillon d'espèce.
- Les mauvaises solutions acceptent les caractéristiques des bonnes solutions pour améliorer leur qualité. Ceci est analogue au mauvais habitat qui accepte l'immigration des espèces depuis d'autres habitats.

La figure 2.1 illustre un modèle d'abondance des espèces dans un habitat. Elle présente des courbes d'immigration et d'émigration comme une ligne droite (modèle simple), ce qui nous donne une description générale du processus d'immigration et d'émigration.
Comme décrit dans l'algorithme de migration (Algorithme 2), supposons que nous ayons une population de solutions candidates à un problème, représentées par des vecteurs (habitats $H_i, i = 1...n$). Chaque élément du vecteur est considéré comme une valeur SIV. Dans le processus de migration, les caractéristiques des bonnes solutions remplacent les mauvaises en utilisant le taux d'immigration λ (2.13) et le taux d'émigration μ (2.14) selon la probabilité de changement P_{mod}.

$$\lambda_k = I(1 - \frac{k}{N}) \tag{2.13}$$

$$\mu_k = E \times \frac{k}{N} \tag{2.14}$$

Où :
λ_k : est le taux d'immigration dans un habitat de k espèces.
μ_k : est le taux d'émigration dans un habitat de k espèces.
E : est le taux maximal d'émigration.
I : est le taux maximal d'immigration.
N : est le nombre maximum d'espèces.
k : est le nombre d'espèces.
Dans le processus de migration (algorithme 2), quand une solution est sélectionnée pour être changée, nous utilisons le taux d'immigration λ pour décider si un SIV sera changé. Dans ce cas, nous utilisons le taux d'émigration μ pour décider quelle bonne solution migrera son SIV. Dans le processus de mutation

Algorithme 2 Algorithme de Migration

Sélectionner un habitat H_i avec une probabilité $\alpha\lambda_i$
si H_i est sélectionné **alors**
  pour $j = 1$ to n **faire**
    Sélectionner un habitat H_j avec une probabilité $\alpha\mu_j$
    si H_j est sélectionné **alors**
      Remplacer un SIV dans H_i avec un SIV de H_j

(algorithme 3), la mutation est exécutée pour toute ou une partie de la population d'une manière similaire à la mutation dans les algorithmes génétiques. Des changements dans un habitat peuvent survenir. Ces modifications changeront le HSI de l'habitat. Ceci est modélisé dans BBO par la mutation avec une certaine probabilité. Le taux de changement de l'habitat est donné dans la formule (2.15).

$$m(s) = \alpha \frac{1 - P_s}{P_{max}} \tag{2.15}$$

Où :
$m(s)$: est le taux de mutation d'un habitat avec s espèces.
α : est un paramètre défini par l'utilisateur
P_s : est la probabilité d'avoir s espèces dans l'habitat (formule 2.16).
P_{max} : est la probabilité d'avoir le nombre maximale d'espèces..
Et :

$$P_k = \begin{cases} \frac{\lambda_0\lambda_1\ldots\lambda_{k-1}}{\mu_1\mu_2\ldots\mu_k(1+\sum_{l=1}^{n}\frac{\lambda_0\lambda_1\ldots\lambda_{l-1}}{\mu_1\mu_2\ldots\mu_l})} & 1 \leqslant k \leqslant n \\ \frac{1}{(1+\sum_{l=1}^{n}\frac{\lambda_0\lambda_1\ldots\lambda_{l-1}}{\mu_1\mu_2\ldots\mu_l})} & k = 0 \end{cases} \tag{2.16}$$

Remarque Les habitats à muter sont ceux qui ont une valeur basse de HSI. Cette mutation introduit la diversité et encourage les habitats moyens à s'améliorer.

Algorithme 3 Algorithme de mutation

pour $j = 1$ à m **faire**
  utiliser λ_i et μ_i pour calculer P_i
  Sélectionner un SIV j de H_i ($H_i(j)$) avec une probabilité αP_i
  si $H_i(j)$ est sélectionné **alors**
    Remplacer $H_i(j)$ par un SIV aléatoire

Dans les stratégies évolutionnaires, la recombinaison est utilisée pour créer de nouvelles solutions, alors que la migration dans BBO est utilisée pour modifier les solutions existantes. La recombinaison globale dans les stratégies évolutionnaires est un processus reproductif, alors que la migration dans BBO est un processus adaptatif ; elle est utilisée pour modifier les habitats existants. Comme dans la plupart des algorithmes d'optimisation à base de population, l'élitisme est typiquement incorporé dans le but de garder la meilleure solution dans la population. Cela empêche que les meilleures solutions soient altérées par l'immigration.
Nous allons dans ce qui suit décrire la première approche développée en implémentant une version améliorée de BBO appelée EBBO.

Algorithme 4 Algorithme BBO

1. Initialiser un ensemble de solutions au problème.
2. Calculer le HSI de chaque solution.
3. Calculer s, λ, μ de chaque solution.
4. Modifier les habitats (migration basée sur λ, μ).
5. Mutation basée sur la probabilité
6. Typiquement, on applique l'élitisme.
7. Répéter à partir de l'étape 2 pour l'itération suivante.

2.3.2 EBBO : Enhanced Biogeography Based Optimization (algorithme d'optimisation amélioré basé sur la biogéographie)

Nous proposons de résoudre HVRPMBTW en utilisant un algorithme amélioré de BBO (Enhanced Biogeography Based Optimization). Cette amélioration est effectuée par l'application de l'algorithme recuit simulé à chaque

solution dans chaque itération. L'approche proposée est présentée dans l'algorithme 5 [Berghida 2014a] .

Algorithme 5 Algorithme EBBO

1. Réglage des paramètres : nombre d'itérations, la taille de la population, la température, α, I, E.
2. Créer un ensemble initial d'habitat satisfaisant les contraintes de problème.
3. Calculer le HSI de chaque habitat.
4. Calculer s, λ, μ de chaque habitat.
5. Modifier les habitats (migration basée sur λ, μ) satisfaisant les contraintes de problème.
6. Mutation basée sur la probabilité et en satisfaisant les contraintes du problème.
7. Appliquer le recuit simulé sur chaque solution.
8. Appliquer l'élitisme
9. Répéter à partir de l'étape 3 pour l'itération suivante.

2.3.2.1 Encodage de la solution

Le codage est l'une des étapes les plus cruciales. En effet, la qualité des résultats peut en dépendre complètement. C'est pourquoi il faut définir le codage le plus adéquat vis-à-vis du problème à résoudre. Nous proposons de coder la solution comme un vecteur de taille k, tel que k représente le nombre total des véhicules. Chaque route est représentée par un véhicule. Chaque élément du vecteur est une liste de clients visités par le véhicule, en commençant par le dépôt de départ 0 et en terminant dans le dépôt d'arrivée $N + 1$.

Exemple Supposant qu'on ait 5 clients (codés de 1 à 5) et 4 véhicules (codés de $V1$ à $V4$). 0 et 6 représentent les noeuds de dépôt : de départ et d'arrivée respectivement. Une solution faisable est représentée dans la table 2.1.

Cette solution utilise seulement trois véhicules, le véhicule 1 visite le client 1, le client 2 ensuite il retourne au dépôt. Le véhicule 3 visite le client 3 ensuite 5 et retourne au dépôt alors que le véhicule 4 visite seulement le client 4. Le véhicule 2 est non utilisé.

2.3.2.2 Description de l'algorithme

Création d'une population initiale d'habitats Dans la première étape, nous générons un ensemble initial d'habitats (algorithme 6). Chaque habitat

V1	V2	V3	V4
0	0	0	0
1	6	3	4
2		5	6
6		6	

TABLE 2.1 – Exemple d'une solution faisable

code une solution possible. Chaque solution (habitat) est générée aléatoirement (solution faisable).

Algorithme 6 Création de la population initiale

Initialiser les routes par le noeud de départ 0 (chaque route r est associée à un véhicule)
répéter
  tant que $(r \leq k), k$ est le nombre de véhicules **faire**
    Sélectionner le plus proche client au client dernièrement ajouté.
    si les contraintes de capacité et de temps de fenêtre sont vérifiées **alors**
      Le client sélectionné est insérée à la tournée r.
jusqu'à Tous les clients sont servis
Toutes les tournées se terminent au dépôt d'arrivée $N + 1$.

Calcul du HSI Après la création de l'ensemble des habitats, chaque habitat est évalué et se voit assigné une valeur de fitness (HSI), selon la fonction de fitness suivante :

$$HSI = \sum_{r=1}^{k} (\text{distance parcourue par la route r} \times \text{coût variable (r)}) \quad (2.17)$$

Calcul de s, λ et μ

- **Calcul de s (nombre d'espèces) :** Les bonnes solutions sont caractérisées par un nombre élevé d'espèces. Pour associer un nombre d'espèces pour chaque solution, nous trions les solutions par ordre décroissant de la valeur du fitness. Nous associons le rang de la solution au nombre d'espèces. Alors la meilleure solution aura le nombre d'espèces le plus élevé et la plus mauvaise le plus bas.
- Calcul de λ (taux d'immigration) et μ (taux d'émigration) Les variables λ et μ caractérisant l'habitabilité sont calculées comme suit :

$$\lambda = I \times (1 - \frac{s}{n}) \quad (2.18)$$

$$\mu = E \times (\frac{s}{n}) \tag{2.19}$$

Tel que I et E sont des constantes qui représentent le taux maximal d'immigration et d'émigration respectivement. n est le nombre total d'habitats.

Migration basée sur λ et μ L'opérateur de migration décrit dans l'algorithme 2, est utilisé pour partager les informations entre les solutions.

Exemple Soient H_1 et H_2 deux solutions générées dans la même itération. Supposons que H_1 est une mauvaise solution sélectionnée pour la migration et H_2 une bonne solution sélectionnée pour améliorer H_1. Les deux solutions sont données dans les tables 2.2, 2.3. Nous décrivons dans ce qui suit le processus

V1	V2	V3	V4
0	0	0	0
5	2	6	4
6	3		6
	1		
	6		

TABLE 2.2 – Habitat H_1

V1	V2	V3	V4
0	0	0	0
6	1	4	3
	6	2	5
		6	6

TABLE 2.3 – Habitat H_2

de migration : nous sélectionnons aléatoirement un SIV de H_1 (supposons que c'est $V2$, on le note $H_1.V_2$). Nous sélectionnons aléatoirement un SIV de H_2 (supposons que c'est V_4, on le note $H_2.V_4$). Si la contrainte de capacité est vérifiée, H_2 partage son SIV avec H_1 (c.à.d. $H_1.V_2 = H_2.V_4$). H_2 devient la solution représentée dans la table 2.4. Nous remarquons que le client 5 est visité par le véhicule 1 et 2, alors que les clients 2 et 1 ne sont pas servis. La solution est donc infaisable. Pour transformer H_1 en une solution faisable, nous proposons d'appliquer le processus suivant :

– Le SIV ayant migré est maintenu.

V1	V2	V3	V4
0	0	0	0
5	3	6	4
6	5		6
	6		

TABLE 2.4 – L'habitat H_1 après la migration

- Les clients déjà visités sont supprimés des autres tournées.
- Les clients manquants sont insérés dans les autres routes en appliquant ce qui suit :
  - Vérifier les contraintes de capacité et de temps de fenêtre et,
  - Trouver la meilleure position d'insertion (en terme de coût). L'insertion du client k entre deux clients i et j, ne change pas le temps d'arrivée chez le client j.

Mutation L'opérateur de mutation a tendance à augmenter la diversité de la population en prospectant d'autres régions de l'espace de recherche. Cet opérateur entraine une perturbation dans l'habitat originel. Le processus de mutation consiste à changer deux SIVs. Cet échange est fait si la contrainte de capacité reste vérifiée.

Exemple Supposons que l'habitat H représenté dans la table 2.5 soit une solution candidate pour la mutation. Supposons que les SIVs aléatoirement sélectionnés pour être échangés dans le processus de mutation soient $V2$ et $V4$. La table 2.6 représente l'habitat après la mutation.

V1	V2	V3	V4
0	0	0	0
6	1	2	6
	5	3	
	4	6	
	6		

TABLE 2.5 – Habitat candidat pour la mutation

Application du recuit simulé Pour améliorer les solutions, nous appliquons le recuit simulé [Kirkpatrick 1983a] pour chaque habitat dans chaque itération. Nous utilisons la méthode 2-opt [Černý 1985a] pour trouver le voisin

V1	V2	V3	V4
0	0	0	0
6	6	2	1
		3	5
		6	4
			6

TABLE 2.6 – Habitat après la mutation

de la solution. Cela améliore le temps d'exécution et évite les conflits dus aux fenêtres de temps. La Figure 2.2 montre le principe de cette méthode.

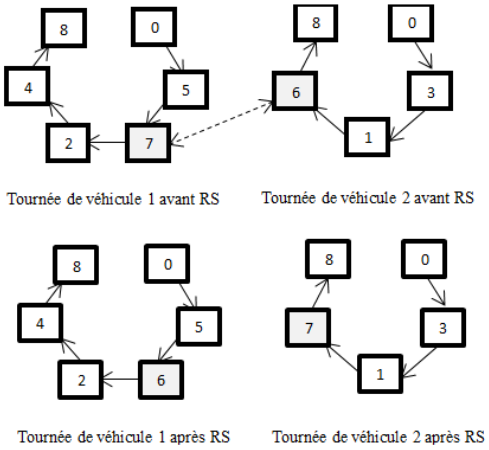

FIGURE 2.2 – Principe 2-opt

Application de l'élitisme Pour éviter de perdre les bonnes solutions après les processus de migration et de mutation. Nous copions une ou plusieurs des meilleures solutions dans la nouvelle génération. Dans notre cas, nous choisissons de copier les deux meilleures solutions.

2.4 Deuxième approche de résolution : Algorithme binaire discret de la recherche de coucous DBCS

2.4.1 CS : Cuckoo search (la recherche de coucous)

Dans la nature, les coucous utilisent une stratégie agressive de reproduction. La femelle investit les nids d'autres oiseaux pour pondre ses oeufs fertilisés [Yang]. Parfois, l'oeuf du coucou dans le nid est découvert et les oiseaux piratés les jettent ou abandonnent le nid et démarrent leur propre couvée ailleurs.
CS est basé sur les trois règles suivantes :

- Chaque coucou pond un oeuf à la fois, et le dépose dans un nid choisi au hasard.
- Les meilleurs nids contenant des oeufs de haute qualité survivent aux prochaines générations.
- Le nombre de nids hôtes disponibles est fixé, et un oiseau hôte peut découvrir un oeuf étranger avec une probabilité $P_a \in [0, 1]$. Dans ce cas, l'oiseau hôte peut soit jeter l'oeuf ou abandonner le nid et construire un nouveau nid dans un nouvel emplacement.

Pour plus de simplicité, la dernière hypothèse peut être approchée par ce qui suit : une fraction Pa des n nids est remplacés par de nouveaux nids [Yang].
Les animaux cherchent la nourriture de façon aléatoire ou quasi-aléatoire. En général, l'emplacement suivant dans le chemin de recherche de nourriture d'un animal dépend de la position actuelle et la probabilité de transition pour le prochain emplacement [Yang]. Diverses études ont montré que le vol de nombreux animaux et insectes présente les caractéristiques typiques de vol de lévy (Lévy Flight) [Brown 2007], [Pavlyukevich 2007], [Reynolds 2007]. Basé sur ces trois règles, les étapes de base de CS sont résumées dans le pseudo-code de l'algorithme 7 [Yang].

Lors de la génération de nouvelles solutions $x_{(t+1)}$ correspondant à un coucou i, un vol de Lévy est réalisé comme suit :

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Lévy}(\lambda) \tag{2.20}$$

Où α est la taille du pas qui dépend de la taille du problème traité. Dans la plupart des cas, on peut utiliser $\alpha = O(L/10)$, où L est la taille caractéristique du problème traité. Le produit $\oplus$ signifie le produit matriciel de Hadamard. Ce produit est similaire à celui utilisé dans PSO. Les Vols de Lévy prévoient essentiellement une marche aléatoire, tandis que leurs pas aléatoires sont tirés

Algorithme 7 Algorithme Cuckoo Search

Fonction objectif $f(x), x = (x_1, ..., x_d)^T$
Générer population initiale de n nids hôtes x_i
tant que $t < MaxGeneration$ ou critère d'arrêt satisfait **faire**
  Obtenir un coucou au hasard / générer une solution x^{t+1} par le vol de Lévy et évaluer sa qualité/fitness F_i
  Choisissez un nid parmi n (par exemple, j) au hasard
  si $F_i > F_j$ **alors**
    Remplacer j par la nouvelle solution
  Une fraction (p_a) des mauvais nids est abandonnée et de nouvelles solutions sont construites
  Garder les meilleures solutions (ou des nids avec des solutions de qualité)

  Trier les solutions et trouver la meilleure solution courante

d'une distribution de Lévy pour les grandes pas [Yang] :

$$\text{Lévy } \tilde{} u = t^{(-\lambda)}, 1 < \lambda \leq 3 \tag{2.21}$$

Cela a une variance infinie avec une moyenne infinie. Ici, les sauts / étapes consécutives d'un coucou forment essentiellement un processus de marche aléatoire [Yang].
Dans la section qui suit, nous allons décrire l'adaptation de CS pour résoudre le HVRPMBTW.

2.4.2 DBCS : Discrete Binary Cuckoo Search (Algorithme discret binaire de recherche de coucous)

Les problèmes d'optimisation binaires discrets sont une classe de problèmes discrets où la solution peut être représentée par un ensemble de bits. Beaucoup de problèmes d'optimisation peuvent être modélisés sous forme binaire, comme le problème d'ordonnancement flow-shop [Liao 2007], les problèmes de routage [Zhan 2009], le MKP [Kong 2006], le KP quadratique [Julstrom 2005]et le quadratique multiples KP [Singh 2007].
L'algorithme CS est basé sur le vol de Lévy, qui produit des nombres réels. Afin de l'adapter au problème HVRPMBTW, nous utilisons la fonction sigmoïde 2.22 pour transformer une solution x_i produite par le vol de Lévy en une solution binaire notée x_i'.

$$S(x_i) = \frac{1}{(1 + e^{-x_i})} \tag{2.22}$$

$x_i^{'}$ est calculé comme suit :

$$x_i^{'} = \begin{cases} 1 & \text{si } r < S(x_i), \text{où } r \text{ est un nombre aléatoire } \in [0,1] \\ 0 & \text{sinon} \end{cases} \tag{2.23}$$

2.4.2.1 Pseudo code de l'algorithme

Le pseudo-code de l'algorithme proposé est mentionné dans l'algorithme 8 [Berghida].

Algorithme 8 Pseudo code de l'algorithme DBCS

La fonction objectif $min \sum_{k \in K} \sum_{j:(i,j) \in A} c_{ij} x_{ij}^k z^k$
Générer une population initiale de n nids hôtes A_i
tant que $t < MaxGeneration$ ou critère d'arrêt satisfait **faire**
  obtenir un coucou C_i aléatoirement :
  Générer une solution par le vol de Lévy (Matrice A_i et $Rang_i$) :
  $A_i^{(t+1)} = A_i^t \oplus \alpha \times \textbf{Lévy}(\lambda), Rang_i^{t+1} = Rang_i^t \oplus \alpha \times \text{Lévy } (\lambda)$
  Obtenir une représentation binaire de A_i en utilisant la fonction sigmoïde

  Vérifier que chaque client est visité par un seul véhicule
  $Rang_i = Rang_i \times A_i$
  Obtenir le nouvel ordre de clients à partir de la matrice $Rang_i$
  Vérifier les contraintes de capacité et de temps de fenêtre
  Évaluer la qualité F_i de C_i
  Choisir un nid parmi les n (par exemple, j) au hasard
  si $F_i > F_j$ **alors**
    Remplacer le nid j par la nouvelle solution
  une fraction (p_a) des mauvais nids est abandonnée et de nouvelles solutions sont construites par le vol de Lévy
  Garder meilleures solutions (ou des nids avec des solutions de qualité)
  Trier les solutions et trouver la meilleure solution courante

2.4.2.2 Encodage de la solution

La solution est représentée par une matrice $A[i,j]$ où $1 \leq i \leq k$ et $1 \leq j \leq n$, tel que k est le nombre total de véhicules ; et n le nombre de clients.

$$A[i,j] = \begin{cases} 1 & \text{Si le véhicule } i \text{ visite le client } j \\ 0 & \text{Sinon} \end{cases} \tag{2.24}$$

Exemple Supposons que nous ayons 5 clients et 4 véhicules ; une solution faisable est représentée par le tableau 2.7. Nous stockons les informations

	Client 1	**Client 2**	**Client 3**	**Client 4**	**Client 5**
Véhicule 1	0	1	1	0	0
Véhicule 2	0	0	0	0	0
Véhicule 3	1	0	0	0	1
Véhicule 4	0	0	0	1	0

TABLE 2.7 – Exemple de la matrice A

concernant l'ordre de passage des véhicules dans une matrice appelée matrice Rang (RM). La matrice de rang associé à la matrice A précédente est donnée dans le tableau 2.8. Cette solution utilise seulement trois véhicules. Le véhicule 1 visite le client 3, le client 2, et retourne au dépôt. Le véhicule 3 visite du client 1, le client 5 et retourne au dépôt, tandis que le véhicule 4 ne visite que le client 4. Le véhicule 2 n'est pas utilisé.

2.5 Troisième approche de résolution : Algorithme amélioré discret de la recherche d'harmonie

2.5.1 HS : Harmony Search (la recherche d'harmonie)

HS est une métaheuristique qui s'inspire du processus d'improvisation dans la musique. Quand un musicien improvise, il a trois options possibles : (1) jouer une harmonie connue de sa mémoire ; (2) jouer quelque chose de semblable à une harmonie connue (pas d'ajustement), ou (3) composer de nouvelles notes aléatoirement. En 2001, Zong Woo Geem et al. ont formalisé ces trois options dans un processus d'optimisation, et les trois composantes correspondant à

	Client 1	**Client 2**	**Client 3**	**Client 4**	**Client 5**
Véhicule 1	0	2	1	0	0
Véhicule 2	0	0	0	0	0
Véhicule 3	1	0	0	0	2
Véhicule 4	0	0	0	1	0

TABLE 2.8 – Exemple de la matrice RM

ces trois options deviennent : considération de la mémoire de l'harmonie, pas d'ajustement, et la randomisation [Geem 2001] . Il utilise l'analogie suivante :

- L'instrument i est analogue à une variable de décision x_i.
- Une note de musique est analogue à la valeur de variable de décision.
- Une harmonie est analogue à une solution.

HS utilise les paramètres suivants :

- HMS **(taille de la mémoire de l'harmonie) :** Ce paramètre représente le nombre d'harmonies (solutions) dans la mémoire de l'harmonie.
- $HMCR$ **(taux de considération de la mémoire d'harmonie) :** Ce paramètre représente la probabilité de générer une nouvelle harmonie de la mémoire de l'harmonie.
- PAR **(taux du pas d'ajustement) :** Ce paramètre représente la probabilité d'ajuster une note, après avoir examiné la mémoire de l'harmonie.

Les principales étapes de l'algorithme HS sont :

1. **Initialisation des paramètres :** Dans cette étape, HMS, $HMCR$ et PAR sont initialisés.
2. **Génération de solutions initiales (initialisation de HM)** : Dans cette étape, un ensemble de solutions sont générées de manière aléatoire et stockées dans HM. Pour chaque solution $i(i = 1...HMS)$, la fonction f_i objectif est calculée. HM peut être considérée comme une matrice contenant un ensemble d'harmonies représentant des solutions.

$$HM = \begin{pmatrix} x_1^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^{HMS} & \cdots & x_n^{HMS} \end{pmatrix} \tag{2.25}$$

3. **Improvisation d'une nouvelle harmonie :** Une nouvelle harmonie $x' = (x'_1, ..., x'_n)$ peut être générée à partir de la mémoire de l'harmonie avec une probabilité $HMCR$ (compte tenu de la mémoire de l'harmonie) ou générée de façon aléatoire avec une probabilité $1 - HMCR$ (randomisation). Dans le premier cas, la valeur de l'i-ème variable de décision dans la nouvelle harmonie x' est choisie à partir des valeurs existantes dans l'i-ème colonne de la mémoire d'harmonie HM courante. Cette valeur peut être ajustée avec la probabilité PAR. Dans le second cas, chaque valeur de la variable de décision est générée de façon aléatoire (à noter que l'ajustement n'est nécessaire que dans le premier cas). La nouvelle harmonie x' remplace la plus mauvaise harmonie dans HM, si elle est meilleure.
4. Si le critère d'arrêt n'est pas satisfait, allez à l'étape 3.

2.5.2 EDHS (Enhanced Discrete Harmony Search)

2.5.2.1 Encodage de la solution

Une solution est représentée par une chaîne d'entiers, où chaque nombre représente le numéro attribué au client servi ou à être servi. Nous utilisons le nombre 0 (représentant le dépôt) comme symbole de séparation des tournées. Comme chaque tournée commence et se termine au dépôt, tout composant de la solution commence et se termine par 0.
Nous avons utilisé un codage direct de la solution (toute information sur la solution est contenue dans la représentation). Dans ce codage, l'ordre de service des clients est reflété directement. L'utilisation d'EDHSA est simplifiée à l'aide de ce codage et les contraintes sont facilement vérifiées.
Une solution est représentée par un vecteur $H = (x_1, x_2, ..., x_k, ..., x_n)$ où x_k représente le nombre assigné à un client ou au dépôt.

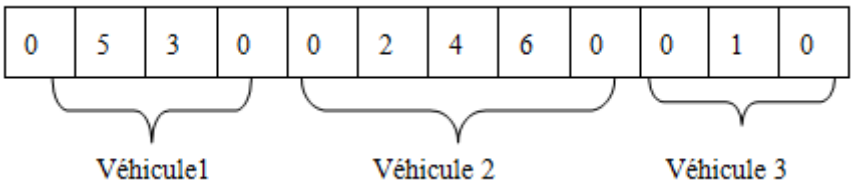

FIGURE 2.3 – Exemple d'une solution

2.5.2.2 Description de l'algorithme

Génération d'une population de solutions nommée mémoire d'harmonies Dans la première étape, nous générons aléatoirement un ensemble initial d'harmonies (Algorithme 6) [Berghida 2014b]. Chaque harmonie encode une solution réalisable.

Improvisation A chaque étape de l'algorithme, nous générons une nouvelle harmonie de deux façons possibles, en fonction de la valeur de $HMCR_i$ générée de façon aléatoire :

1. $HMCR_i < HMCR$ **:** Dans ce cas, la nouvelle solution est générée à partir de la mémoire d'harmonie. En effet, pour chaque x_k^{new} de l'harmonie générée (qui représente le client k à visiter), nous prenons une solution $S_t, t \in [1, HMS]$ au hasard de la mémoire puis, en fonction de la valeur du paramètre PAR, nous avons deux cas possibles :

- $PAR_i < PAR$: Nous faisons un ajustement de la valeur générée par le voisinage de la valeur x_k^t dans la population : soit nous prenons la valeur $x_k^{(t-1)}$ de la solution S_{t-1} soit la valeur $x_k^{(t+1)}$ de la solution S_{t+1}.
- $PAR_i > PAR$ Nous considérons la valeur x_k^t de la solution S_t. Ensuite, nous vérifions les contraintes de le la nouvelle harmonie.

2. $HMCR_i > HMCR$: Dans ce cas, et pour chaque x_k^{new} de la nouvelle harmonie, nous choisissons au hasard un client de la liste des clients disponibles. La solution produite doit satisfaire les contraintes. Après avoir généré la nouvelle harmonie, nous appliquons le recuit simulé pour améliorer la qualité de la solution

Mise à jour de la mémoire d'harmonie Si la qualité de la nouvelle harmonie est meilleure que la qualité de la pire harmonie dans HM, nous remplaçons la pire harmonie par la nouvelle.

2.6 Quatrième approche de résolution : Algorithme quantique de recherche d'harmonie avec taille de population variable

2.6.1 QC : Quantum Computing (l'informatique quantique)

L'informatique quantique QC est une nouvelle théorie qui a émergé à la suite de la fusion de l'informatique et de la mécanique quantique. Elle s'appuie sur les principes de la mécanique quantique comme la représentation et superposition d'états Q-bit. QC est capable de traiter un très grand nombre d'états quantiques simultanément en parallèle. Ce parallélisme réduit évidemment la complexité algorithmique, et peut être utilisé pour résoudre des problèmes d'optimisation combinatoire, qui nécessitent l'exploration de grands espaces de solutions.
Les algorithmes quantiques ne peuvent pas être exploités avant la construction de machines puissantes quantiques. Ils nécessitent la présence de machines quantiques qui prennent en charge la représentation quantique grâce à l'utilisation de grilles quantiques. En attendant que ces machines soient construites, une utilisation intermédiaire a émergé. C'est l'informatique inspirée du quantique (Quantum Inspired Computing).
QIA (Quantum Inspired Algorithms) sont de nouveaux algorithmes opérant sur des machines conventionnelles et enrichis par les concepts de l'informatique quantique. Ils exploitent les principes quantiques afin de réduire la com-

plexité algorithmique nécessaire pour résoudre des problèmes complexes. La particularité de QIA vient de la représentation quantique qu'ils adoptent, qui permet de représenter la superposition de toutes les solutions possibles pour un problème donné.

Concepts basiques de QIA

- Le Q-bit est la plus petite unité d'information stockée dans un ordinateur.
- Un individu avec une chaîne de m Q-bits peut être exprimé comme suit : $\begin{bmatrix} \alpha_1 & \alpha_1 & ... & \alpha_m \\ \beta_1 & \beta_2 & ... & \beta_m \end{bmatrix}$ Où :

$$|\alpha^2|+|\beta^2|= 1 \tag{2.26}$$

- Un Q-bit peut être dans l'état 1, ou dans l'état 0 ou dans n'importe quelle superposition de ces deux valeurs (α et β sont des nombres complexes).
- Lorsque nous mesurons l'état du Q-bit, nous pouvons avoir 0 avec une probabilité $|\alpha^2|$, et 1 avec une probabilité de $|\beta^2|$.
- QIA consiste à appliquer successivement une série d'opérations quantiques sur un système quantique (mesure, mutation et interférence). ces trois opérations seront détaillés dans les sections qui suivent.

2.6.2 QIHSVPS : Quantum Inspired Harmony Search algorithm with Variable Population Size

Nous proposons d'utiliser un algorithme inspiré de l'informatique quantique avec une taille de population variable pour résoudre le problème de HVRPMBTW.
Nous avons choisi de réduire la taille de la population lorsque l'amélioration de la qualité de la solution est insignifiante. La réduction de la taille de la population a été utilisé en [Minetti 2005], [Chen 2012b] et [Chen 2010].
Les principes de l'informatique quantique sont utilisés pour accélérer le processus d'évolution alors que la taille de la population variable est utilisée pour diminuer le nombre d'évaluations de la solution, lorsque l'amélioration est négligeable. L'algorithme 9 présente le pseudo-code de l'approche proposée.

2.6.2.1 La représentation quantique de l'harmonie

La représentation en Q-bits dans les QIA explore seulement les problèmes ayant des solutions présentées en binaire (espace de recherche $(0-1)$), tandis que la solution dans le VRP est une permutation de tous les clients.
Pour résoudre ce problème, nous développons une méthode de codage qui

Algorithme 9 Pseudo Code of QIHSVPS

Créer une population initiale HM des harmonies
Trier la population HM
Appliquer l'élitisme
tant que $t < NbrIter$ et $HM.size \geqslant 1$ **faire**
  $PAR = ((PAR_{max} - PAR_{min})/NbrIter) \times t$
  Improviser une nouvelle harmonie H_{new}
  Appliquer la mesure
  si $Fitness(H_{new}) > WorstFitness$ **alors**
    Enlever la pire harmonie et la remplacer par la nouvelle H_{new}
  Appliquer l'élitisme
  pour $i = 0$ à $HM.size$ **faire**
    si $r \in [0, 1] < ProbMutation$ **alors**
      Appliquer la mutation inter Q-bit à H_i
    sinon
      Appliquer la mutation intra Q-bit à H_i
    Appliquer l'interférence to H_i
  trier la population HM et obtenir la meilleure solution courante (t)
  $RateImprov = (F(BH(t-1)) - F(BH(t)))/(F(BH(t))) \times 100\%$
  si $RateImprov > 0.05$ **alors**
    copier les deux meilleures solution à la génération suivante
    Une fraction d'harmonies est supprimée du HM en utilisant la sélection par tournoi

convertit la solution présentée en Q-bits à une solution contenant des nombres entiers. Une solution est représentée par un vecteur de Q-bits de taille $l = n(k+m)$, où m est le nombre total de clients, n est le nombre de bits nécessaires pour coder le nombre $(m+1)$, et k est le nombre total de véhicules. Le vecteur de Q-bit est ensuite converti en binaire en utilisant les valeurs de α et β. Ce dernier est ensuite converti en un vecteur de nombre entier de la manière suivante :

- Le vecteur binaire est divisé en $(k + m)$ chaîne ; chaque chaîne binaire de taille n est ensuite converti en entier.
- Les nombres entiers sont ensuite triés, les k petits nombres seront remplacés par un zéro. Les m nombres restants seront numérotés de 1 à m, chacun représentant le numéro de client.
- Nous vérifions ensuite si les contraintes de capacité et de fenêtres de temps sont satisfaites. Si ce n'est pas le cas, la solution obtenue est transformée en une solution réalisable.

Exemple Supposons que nous ayons 5 clients et 4 véhicules. Une représentation en Q-bits de la solution est un vecteur de 27 éléments. La figure 2.4 illustre le processus d'encodage. La solution utilise 3 véhicules. Le véhicule 2 visite le client 1 ensuite le client 2. Le véhicule 3 visite le client 5 ensuite le client 4. Tandis que le véhicule 4 visite le client 3. Le véhicule 1 n'est pas utilisé.

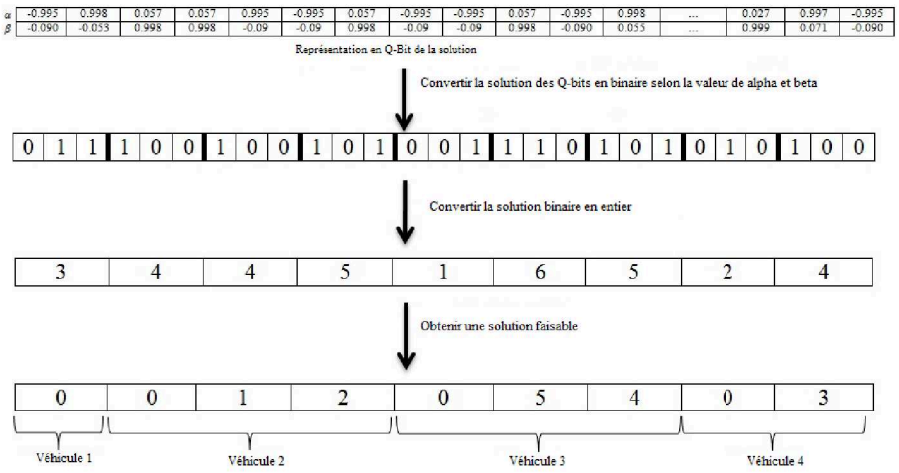

FIGURE 2.4 – Processus d'encodage

2.6.2.2 Description de l'algorithme

Création de la population initiale des harmonies Dans la première étape, nous générons aléatoirement un ensemble initial d'harmonies. Chaque harmonie est un vecteur de taille $l = n(k+m)$, tel que n est le nombre de bits nécessaires pour coder le nombre $(m+1)$, k est le nombre total de véhicules, m est le nombre total de clients.

Improvisation d'une nouvelle harmonie A chaque itération, une nouvelle harmonie est générée suivant l'algorithme 10

La mesure (Measurement) La mesure consiste à transformer par projection le vecteur quantique en un vecteur binaire (Figure 2.5). Pour chaque Q-bit, nous générons un nombre aléatoire $\delta \in [0,1]$: Si $(|\alpha^2| > \delta)$, nous mettons le bit à 1 sinon nous le mettons à 0. A la fin nous obtenons un vecteur

Algorithme 10 Improviser une nouvelle harmonie

Une harmonie vide H_{new} est créée, qui est un vecteur de $l = n \times (k + m)$ Q-bit (tel que n est le nombre de bits requis pour coder $(m + 1)$, k est le nombre total de véhicules, m est le nombre total des clients)
tant que $i < l$ **faire**
  si $r \in [0, 1] < HMCR$ **alors**
    choisir une harmonie à partir de HM par sélection par roulette
    si $r \in [0, 1] < PAR$ **alors**
      Choisir au hasard deux Q-bits de tournée actuelle et échanger leur emplacement
    sinon
      Copier la route telle qu'elle est à H_{new}
  sinon
    Générer une route vide, et lui attribuer un ensemble aléatoire de Q-bits.
  $i = i + SizeOfRoute$

binaire. Ce vecteur binaire sera converti en un entier, en utilisant la procédure de codage décrit dans la section (**La représentation quantique de l'harmonie**).

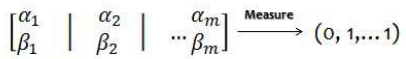

FIGURE 2.5 – Mesurer une solution

La mutation La mutation permet le déplacement de la solution actuelle à une de ses voisins, nous distinguons deux types de mutations quantiques :

- **La mutation Inter-Q-bits**, qui consiste à permuter deux Q-bits choisis aléatoirement appartenant à une harmonie choisie aléatoirement.
- **La mutation intra-Q-bits**, qui consiste à permuter les amplitudes α et β d'un Q-bits choisi aléatoirement.

L'interférence L'interférence est un opérateur d'évolution qui guide les individus vers de meilleures solutions, et finalement, vers un seul état. Le $[\alpha_i, \beta_i]^T$ d'i-ème Q-bit est mise à jour comme suit :

$$\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = U(\theta_i) \times \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \times \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{2.27}$$

Tel que :

– θ_i est l'ongle de rotation, $\theta_i = s(\alpha_i, \beta_i)\Delta\theta_i$.
– $s(\alpha_i, \beta_i)$ est le signe de θ_i qui détermine la direction de la rotation, et
– $\Delta\theta_i$ est l'amplitude de l'angle de rotation.

Nous avons utilisé une valeur dynamique de l'angle de rotation afin d'accroître la performance de l'opérateur d'interférence. Les valeurs de rotation d'angle sont indiquées dans le tableau 2.9 [Zhang 2009a]. r_i et b_i sont le i-ème bit de l'harmonie binaire r et le i-ème bit de la meilleure harmonie b respectivement.

r_i	b_i	$f(r_i) < (b_i)$	$\Delta\theta_i$	$s(\alpha_i, \beta_i)$			
				$\alpha\beta_i > 0$	$\alpha\beta_i < 0$	$\alpha = 0$	$\beta = 0$
0	0	false	0	0	0	0	
0	1	true	0	0	0	0	0
0	0	false	0	0	0	0	0
0	1	true	0.05π	-1	+1	+1	0
1	0	false	0.01π	-1	+1	+1	0
1	1	true	0.025π	+1	-1	0	+1
1	0	false	0.005π	+1	-1	0	+1
1	1	true	0.025π	+1	-1	0	+1

TABLE 2.9 – Lookup de rotation d'angle

2.7 Résultats des approches EBBO, DBCS, EDHS, QIHSVPS sur le problème HVRPMBTW

2.7.1 Jeux de données utilisé

Nous testons nos approches sur différents ensembles de problème dérivés de l'ensemble de données de Solomon [Solomon 1987] $R1, R2, C1, C2, RC1$ et $RC2$. Ces données contiennent 55 instances de VRPTW, chacune contient 100 clients. Chaque ensemble de données contient entre huit et douze problème de 100 noeuds. Pour $R1$ et $R2$, les clients sont localisés d'une manière uniforme sur la zone de service. Les ensembles $C1$ et $C2$ sont des clients regroupés et les groupes $RC1$ et $RC2$ sont une combinaison des clients regroupés et des clients placés aléatoirement. Tous les problèmes sont euclidiens avec un seul type de service.
Dans le but de s'accorder avec le problème HVRPMBTW, Belmecheri et al. [Belmecheri 2013] ont modifié les jeux de données comme suit : ils considèrent

que chaque instance a un pourcentage de 80% des clients de livraison et 20% des clients de collecte. Les flottes hétérogènes fixes proposées pour $n = 5, 100$ clients sont présentées dans les tables 2.10 et 2.11 respectivement. Pour comparer les résultats de nos approches avec ceux de [Belmecheri 2013], nous utilisons les mêmes modifications.

Type de véhicule	Nombre de véhicules	coût variable	Capacité
a	2	1.2	50
b	2	1.0	40

TABLE 2.10 – Flotte fixe hétérogène pour $n = 5$ clients

Type de véhicule	Nombre de véhicules	Coût variable	Capacité
a	5	2.0	600
b	5	1.8	400
c	8	1.6	200
d	6	1.5	100
e	6	1.0	50

TABLE 2.11 – Flotte fixe hétérogène pour $n = 100$ clients

2.7.2 Comparaison de EBBO, DBCS, EDHS, QIHSVPS avec une méthode exacte

Nous commençons les expériences en comparant les résultats des quatre approches avec les résultats obtenus par une méthode exacte pour les instances de petites taille (5 clients). les résultats exacts sont trouvés en utilisant LPSolve IDE. le tableau 2.12 montre que les approches proposées donnent des résultats proches de l'optimal (Nb représente le nombre d'instances, EM représente les résultats obtenus par la méthode exacte). Cela est dû au fait que l'espace de recherche est réduit.

2.7.3 Comparaison de EBBO, DBCS, EDHS, QIHSVPS avec PSO (Particle Swarm Optimization) et ACO (Ant Colony Optimization)

Dans cette section, nous comparons chacune des quatre approches proposées avec PSO et ACO à l'aide de l'opérateur GAP qui calcul l'écart entre

Instance	Nb	EM	EBBO	EDHS	DBCS	QIHSVPS
R1-5	12	152.66	154.45	156.32	154.45	152.66
R2-5	11	145.19	145.19	148.84	145.19	145.19
C1-5	9	77.45	77.54	77.54	77.54	77.54
C2-5	8	110.19	110.19	110.19	110.19	110.19

TABLE 2.12 – Solution moyenne de 5 clients

deux solution, et qui est défini comme suit :

$$GAP_{Approche1-Approche2} = \frac{Cost_{Approche1} - Cost_{Approche2}}{Cost_{Approche2}} \times 100\% \qquad (2.28)$$

Les résultats sont résumés dans les tableaux 2.13, 2.14, 2.15 et 2.16. La première colonne représente le nom de l'instance, les résultats obtenus par ACO et PSO [Belmecheri 2013] sont présentés dans la deuxième et la troisième colonne respectivement, les résultats des approches proposées EBBO, DBCS, EDHS et QIHSVPS sont donnés dans la quatrième colonne dans le tableau 2.13, 2.14, 2.15 et 2.16 respectivement . Les deux dernières colonnes représentent les écarts entre les solutions trouvées par (ACO et l'approche proposée) et (PSO et l'approche proposée) respectivement.

A partir des tableaux 2.13,2.14,2.15 et 2.16 nous remarquons que :

- Dans le premier type d'instances (C1, C2), où les clients sont regroupés ; EBBO améliore les résultats obtenus par ACO avec une moyenne de 5%, et améliore ceux des PSO avec $8,5\%$. DBCS améliore aussi les deux approches avec 2.71% , 1.7% pour ACO, PSO respectivement. L'approche EDHS n'a pas amélioré les résultats des deux approches, alors que l'approche QIHSVPS a amélioré les résultats d'ACO avec une moyenne de 1.75% et les résultats de PSO avec une moyenne de 10.88%.
- Dans le deuxième type d'instances (R1, R2), où les clients sont distribués normalement, PSO donne une meilleure moyenne qu'EBBO ,ce dernier améliore les résultats d'ACO avec $1,175\%$. L'approche DBCS a amélioré les résultats d'ACO et PSO avec une moyenne de 2.79%, 0.46%respectivement. EDHS n'a pas amélioré en moyenne ces instances. Tandis que l'approche QIHSVPS a amélioré les résultats d'ACO et PSO avec 2.86%, 1.19% respectivement.
- Dans le dernier type d'instances (RC1, RC2), EBBO améliore les résultats de PSO et ACO avec une moyenne de $4,08\%$, $7,83\%$ respectivement. DBCS les améliore aussi avec $0.05\%, 0.01\%$ respectivement. l'approche EDHS a amélioré les résultats d'ACO avec une moyenne de 6.07% et les résultats de PSO avec une moyenne de 1.84%. Alors que l'approche

Instance	ACO	PSO	EBBO	GAP_1 %	GAP_2 %
C101-100	2,892.92	2,560.02	2,331.54	24.07	9.79
C102-100	2,890.61	2,615.32	2,410.18	19.93	8.51
C103-100	2,502.23	2,405.3	2,102.41	19.01	14.40
C104-100	2,204.23	2,333.95	2,021.55	9.03	15.45
C105-100	2,141.82	2,055.9	1998.83	7.15	2.85
C106-100	2,298.4	2,366.05	2,193.54	4.78	7.86
C107-100	2,035.59	1,992.83	2,188.36	-6.98	-8.93
C108-100	1,963.67	1,938.54	1,950.36	0.68	-0.60
C109-100	2,048.73	2,234.79	1,786.66	14.66	25.08
Moyenne				**10.26**	**8.27**
C201-100	1,298.17	1,420.62	1326.76	-2.15	7.07
C202-100	1,438.4	1,590.2	1,327.33	8.36	19.80
C203-100	1,564.12	1,823.84	1432.36	9.19	27.33
C204-100	1,727.46	1,856.26	1,884.86	-9.11	-1.51
C205-100	1,354.02	1,504.98	1,296.01	4.47	16.12
C206-100	1,444.86	1,528.31	1687.04	-14.35	-9.40
C207-100	1,376.64	1,391.91	1,416.27	-2.79	-1.72
C208-100	1,515.57	1,626.96	1453.05	4.30	11.96
Moyenne				**-0.26**	**8.74**
Moyenne C1-C2				**5**	**8.5**
R101-100	2,854.82	2,632.13	2567.14	11.20	2.53
R102-100	2671.04	2,375.41	2369.30	12.73	-0.25
R103-100	2,124.03	2,006.8	2080.34	2.10	-3.53
R104-100	1,863.78	1,853.21	1,971.83	-5.47	-6.01
R105-100	2,379.08	2,253.42	2,334.56	1.90	-3.47
R106-100	2,063.28	2,031.19	2,121.66	-2.75	-4.26
R107-100	1,905.43	1,928.9	1,943.65	-1.96	-0.75
R108-100	1,918.02	1,877.52	2,002.36	-4.21	-6.23
R109-100	2,133.42	2,001.56	2,069.38	3.09	-3.27
R110-100	1,979.53	1,983.98	2,065.76	-4.17	-3.95
R111-100	1,889.19	1,896.7	1881.21	0.42	0.82
R112-100	1,936.93	1,895.77	1,689.12	14.76	12.23
Moyenne				**2.30**	**-1.34**
R201-100	2,012.12	1,990.47	1,344.47	49.65	48.04
R202-100	1,922.72	1,932.74	1,941.04	-0.94	-0.42
R203-100	1,736.2	1,754.37	1910.74	-9.13	-8.18
R204-100	1,584.13	1,522.5	1,876.82	-15.59	-18.87
R205-100	1,848.66	1,885.75	1753.49	5.38	7.54
R206-100	1,758.78	1,813.48	1792.46	-1.87	1.17
R207-100	1,650.12	1,654.84	1806.25	-8.64	-8.38
R208-100	1,536.68	1,589.42	1737.67	-11.57	-8.53
R209-100	1,777.49	1,729.58	1798.76	-1.18	-3.84
R210-100	1,793.64	1,754.44	1868.55	-4.00	-6.10
R211-100	1,615.85	1,699.39	1641.38	-1.55	3.53
Moyenne				**0.05**	**0.54**
Moyenne R1-R2				**1.175**	**-0.4**
RC101-100	3.348.18	2.957.49	2387.96	40.21	23.85

Instance	ACO	PSO	DBCS	$GAP_{ACO-DBCS}$ %	$GAP_{PSO-DBCS}$ %
C101-100	2,892.92	2,560.02	2,230.43	29.7	14.77
C102-100	2,890.61	2,615.32	2,271.12	27.27	15.15
C103-100	2,502.23	2,405.3	2,139.97	16.92	12.39
C104-100	2,204.23	2,333.95	2,113.64	4.28	10.42
C105-100	2,141.82	2,055.9	2,147.52	-0.26	-4.26
C106-100	2,298.4	2,366.05	2,676.55	-14.12	-11.6
C107-100	2,035.59	1,992.83	2,580.28	-21.11	-21.94
C108-100	1,963.67	1,938.54	2,046.13	-4.03	-5.25
C109-100	2,048.73	2,234.79	1,839.72	11.36	21.47
Moyenne				5.55	3.46
C201-100	1,298.17	1,420.62	1,845.80	-0.29	-0,23
C202-100	1,438.4	1,590.2	1,952.15	-0.26	-0.18
C203-100	1,564.12	1,823.84	1,780.78	-0.12	0.02
C204-100	1,727.46	1,856.26	1,685.12	0.02	0.1
C205-100	1,354.02	1,504.98	1,368.18	-0.01	0.09
C206-100	1,444.86	1,528.31	1,815.89	-0.2	-0.15
C207-100	1,376.64	1,391.91	1,514.17	-0.09	-0.08
C208-100	1,515.57	1,626.96	1,588.98	-0.04	0.02
Moyenne				**-0.12**	**-0.05**
Moyenne C1.C2				**2.71**	**1.7**
R101-100	2,854.82	2,632.13	2,611.75	9.3	0.78
R102-100	2,671.04	2,375.41	2,405.52	11.03	-1.25
R103-100	2,124.03	2,006.8	1,259.00	68.7	59.39
R104-100	1,863.78	1,853.21	2,001.39	-6.87	-7.4
R105-100	2,379.08	2,253.42	2,348.54	1.3	-4.05
R106-100	2,063.28	2,031.19	2,230.26	-7.48	-8.92
R107-100	1,905.43	1,928.9	2,146.78	-0.11	-10.14
R108-100	1,918.02	1,877.52	2,022.66	-5.17	-7.17
R109-100	2,133.42	2,001.56	2,069.38	3.09	-2.79
R110-100	1,979.53	1,983.98	2,215.58	-10.65	-10.45
R111-100	1,889.19	1,896.7	2,087.10	-9.48	-9.12
R112-100	1,936.93	1,895.77	1,712.08	13.13	10.72
Moyenne				5.56	0.89
R201-100	2,012.12	1,990.47	1,875.45	0.07	0.06
R202-100	1,922.72	1,932.74	1,887.39	0.01	0.02
R203-100	1,736.2	1,754.37	1,800.54	-0.03	-0.02
R204-100	1,584.13	1,522.5	1,658.67	-0.04	-0.08
R205-100	1,848.66	1,885.75	1,714.08	0.07	0.1
R206-100	1,758.78	1,813.48	1,708.44	0.02	0.06
R207-100	1,650.12	1,654.84	1,496.71	0.10	0.10
R208-100	1,536.68	1,589.42	1,462.18	0.05	0.08
R209-100	1,777.49	1,729.58	1,612.78	0.1	0.07
R210-100	1,793.64	1,754.44	1,876.19	-0.04	-0.06
R211-100	1,615.85	1,699.39	1,512.16	0.06	0.12
Moyenne				**0.03**	**0.03**
Moyenne R1.R2				**2.79**	**0.46**
RC101-100	3,348.18	2,957.49	2,845.12	0.17	0.03
RC102-100	2,867.89	2,464.51	2,612.56	0.09	-0.05
RC103-100	2,432.23	2,426.88	2,812.49	-0.13	-0.13
RC104-100	2,400.72	2,244.58	2,142.95	0.12	0.04
RC105-100	3,005.21	2,711.05	2,425.14	0.23	0.11
RC106-100	2,706.68	2,495.57	2,612.80	0.03	-0.04
RC107-100	2,414.86	2,420.42	2,312.45	0.04	0.04
RC108-100	2,444.32	2,381.45	2,214.79	0.1	0.07
Moyenne				**0.08**	**0.008**
RC201-100	2,484.64	2,401.11	2,332.54	0.06	0.02
RC202-100	2,190.48	2,251.39	2,123.68	0.03	0.06
RC203-100	1,931.69	2,022.9	1,892.14	0.02	0,06
RC204-100	1,750.12	1,827.48	1,952.32	-0.1	-0.06
RC205-100	2,226.16	2,274.91	2,349.80	-0.05	-0.03
RC206-100	2,054.41	2,084.5	2,012.12	0.02	0.03
RC207-100	2,012.44	1,836.63	1,725.39	0.16	0.06
Moyenne				**0.02**	**0.02**
Moyenne RC1.RC2				**0.05**	**0.01**

TABLE 2.14 – Comparaison de 100 clients (C1, C2, R1, R2, RC1, RC2) pour l'approche DBCS

Instance	ACO	PSO	EDHS	$GAP_{ACO-EDHS}$ %	$GAP_{PSO-EDHS}$ %
C101-100	2,892.92	2,560.02	2,162.00	33,77	18,40
C102-100	2,890.61	2,615.32	2,256.98	28,07	15,87
C103-100	2,502.23	2,405.3	2,437.24	2,66	-1,31
C104-100	2,204.23	2,333.95	1,994.78	12,30	17,00
C105-100	2,141.82	2,055.9	2,244.86	-4,59	-8,41
C106-100	2,298.4	2,366.05	2,282.00	0,71	3,68
C107-100	2,035.59	1,992.83	2,292.02	-11,18	-13,05
C108-100	1,963.67	1,938.54	2,206.04	-10,98	-12,12
C109-100	2,048.73	2,234.79	2,107.34	-2,78	6,04
Moyenne				**5,33**	**2,90**
C201-100	1,298.17	1,420.62	2,337.97	-44,47	-39,23
C202-100	1,438.4	1,590.2	2,100.57	-31,52	-24,29
C203-100	1,564.12	1,823.84	2,180.00	-28,25	-16,33
C204-100	1,727.46	1,856.26	1,974.98	-12,53	-6,01
C205-100	1,354.02	1,504.98	2,147.94	-36,96	-29,93
C206-100	1,444.86	1,528.31	2,140.30	-32,49	-28,59
C207-100	1,376.64	1,391.91	2,122.82	-35,15	-34,43
C208-100	1,515.57	1,626.96	2,111.57	-28,22	-22,95
Moyenne				**-31,2**	**-25,22**
Moyenne C1.C2				**-12.93**	**-11.16**
R101-100	2,854.82	2,632.13	2,139.72	33,42	23,01
R102-100	2671.04	2,375.41	2,158.78	23,72	10,03
R103-100	2,124.03	2,006.8	1,934.51	9,79	3,73
R104-100	1,863.78	1,853.21	2,011.23	-7,33	-7,85
R105-100	2,379.08	2,253.42	2,134.12	11,47	5,59
R106-100	2,063.28	2,031.19	2,272.33	-9,19	-10,61
R107-100	1,905.43	1,928.9	1,807.51	5,41	6,71
R108-100	1,918.02	1,877.52	2,061.45	-6,95	-8,92
R109-100	2,133.42	2,001.56	1,931.17	10,47	3,64
R110-100	1,979.53	1,983.98	1,891.88	4,63	4,86
R111-100	1,889.19	1,896.7	1,884.17	0,26	0,66
R112-100	1,936.93	1,895.77	2,047.42	-5,39	-7,40
Moyenne				**5.86**	**1.95**
R201-100	2,012.12	1,990.47	2,515.23	-20,00	-20,86
R202-100	1,922.72	1,932.74	2,287.65	-15,95	-15,51
R203-100	1,736.2	1,754.37	2,105.34	-17,53	-17,09
R204-100	1,584.13	1,522.5	1,960.73	-19,20	-20,82
R205-100	1,848.66	1,885.75	2,155.01	-14,21	-12,49
R206-100	1,758.78	1,813.48	2,217.49	-20,68	-18,21
R207-100	1,650.12	1,654.84	2,012.60	-18,01	-17,77
R208-100	1,536.68	1,589.42	1,843.11	-16,62	-13,76
R209-100	1,777.49	1,729.58	2,129.54	-16,53	-18,78
R210-100	1,793.64	1,754.44	2,171.03	-17,38	-19,18
R211-100	1,615.85	1,699.39	1,867.03	-13,45	-8,97
Moyenne				**-17.23**	**-16.68**
Moyenne R1.R2				**-5.68**	**-7.36**
RC101-100	3,348.18	2,957.49	2,761.24	33.17	17.63
RC102-100	2,867.89	2,464.51	2,461.32	27.7	9.73
RC103-100	2,432.23	2,426.88	2,552.63	5.19	4.96
RC104-100	2400.72	2,244.58	2,522.63	6.52	9.87
RC105-100	3,005.21	2,711.05	2,214.51	17.51	3.75
RC106-100	2,706.68	2,495.57	2,475.54	-0.22	-8
RC107-100	2,414.86	2,420.42	2,631.31	3.71	3.95
RC108-100	2,444.32	2,381.45	2,428.89	0.86	-1.73
Moyenne				**12.86**	**5.02**
RC201-100	2,484.64	2,401.11	2,164.7	14,77	10,92
RC202-100	2,190.48	2,251.39	2,292.55	-4,45	-1,79
RC203-100	1,931.69	2,022.9	2,002.99	-3,55	0,99
RC204-100	1,750.12	1,827.48	2,116.22	-17,29	-13,65
RC205-100	2,226.16	2,274.91	2,180.13	0,76	-2,61
RC206-100	2,054.41	2,084.5	2,113.76	-2,80	-1,38
RC207-100	2,012.44	1,836.63	1,872.28	7,48	-1,90
Moyenne				**-0.72**	**-1.34**
Moyenne RC1.RC2				**6.07**	**1.84**

TABLE 2.15 – Comparaison de 100 clients (C1, C2, R1, R2, RC1, RC2) pour l'approche EDHS

Instance	ACO	PSO	QIHSVPS	$GAP_{ACO-QIHSVPS}$ %	$GAP_{PSO-QIHSVPS}$ %
C101-100	2,892.92	2,560.02	2,242.98	28.97	14.13
C102-100	2,890.61	2,615.32	2,523.62	14.54	3.63
C103-100	2,502.23	2,405.3	2,432.14	2.88	-1.1
C104-100	2,204.23	2,333.95	1,952.12	12.91	19.55
C105-100	2,141.82	2,055.9	1,875.48	14.2	9.61
C106-100	2,298.4	2,366.05	2,169.5	5.94	9.06
C107-100	2,035.59	1,992.83	1,812.47	12.31	9.95
C108-100	1,963.67	1,938.54	2,013.31	-2.52	-3.71
C109-100	2,048.73	2,234.79	1,954.22	4.83	14.35
Moyenne				**10.45**	**8.38**
C201-100	1,298.17	1,420.62	1816.65	-14.4	-6.33
C202-100	1,438.4	1,590.2	1,218.44	18.05	30.51
C203-100	1,564.12	1,823.84	1,319.05	18.57	38.26
C204-100	1,727.46	1,856.26	1,612.13	7.15	15.14
C205-100	1,354.02	1,504.98	1,392.17	-2.74	8.1
C206-100	1,444.86	1,528.31	1,1412.24	2.3	8.21
C207-100	1,376.64	1,391.91	1,547.69	-11.05	-10.06
C208-100	1,515.57	1,626.96	1,576.14	-3.84	3.22
Moyenne				**1.75**	**10.88**
Moyenne C1.C2				**6.1**	**9.63**
R101-100	2,854.82	2,632.13	2,643.45	7.99	-0.42
R102-100	2671.04	2,375.41	2,242.67	19.1	5.91
R103-100	2,124.03	2,006.8	2,120.12	0.18	-5.34
R104-100	1,863.78	1,853.21	1,840.23	1.27	0.7
R105-100	2,379.08	2,253.42	2,125.45	11.93	6.02
R106-100	2,063.28	2,031.19	2,002.43	3.03	1.43
R107-100	1,905.43	1,928.9	2,012.8	-5.33	-4.16
R108-100	1,918.02	1,877.52	1,798.42	6.6	4.39
R109-100	2,133.42	2,001.56	1,952.26	9.27	2.52
R110-100	1,979.53	1,983.98	2,042.47	-3.08	-2.86
R111-100	1,889.19	1,896.7	1,723.19	9.63	10.06
R112-100	1,936.93	1,895.77	1,932.12	0.24	-1.8
Moyenne				**5.07**	**1.37**
R201-100	2,012.12	1,990.47	1,994.18	0.89	-0.18
R202-100	1,922.72	1,932.74	1,886.32	1.92	2.46
R203-100	1,736.2	1,754.37	1,812.08	-4.18	-3.18
R204-100	1,584.13	1,522.5	1,444.12	9.69	5.42
R205-100	1,848.66	1,885.75	1,903.18	-2.86	-0.91
R206-100	1,758.78	1,813.48	1,912.06	-8.01	-5.15
R207-100	1,650.12	1,654.84	1,513.13	9.05	9.36
R208-100	1,536.68	1,589.42	1,717.23	-10.51	-7.44
R209-100	1,777.49	1,729.58	1,605.75	10.69	7.71
R210-100	1,793.64	1,754.44	1,665.12	7.71	5.36
R211-100	1,615.85	1,699.39	1,738.72	-7.06	-2.26
Moyenne				**0.66**	**1.01**
Moyenne R1.R2				**2.86**	**1.19**
RC101-100	3,348.18	2,957.49	2,514.21	33.17	17.63
RC102-100	2,867.89	2,464.51	2,245.78	27.7	9.73
RC103-100	2,432.23	2,426.88	2,312.17	5.19	4.96
RC104-100	2400.72	2,244.58	2,042.89	6.52	9.87
RC105-100	3,005.21	2,711.05	2,612.88	17.51	3.75
RC106-100	2,706.68	2,495.57	2,712.66	-0.22	-8
RC107-100	2,414.86	2,420.42	2,328.42	3.71	3.95
RC108-100	2,444.32	2,381.45	2,423.39	0.86	-1.73
Moyenne				**12.86**	**5.02**
RC201-100	2,484.64	2,401.11	2,145.78	15.79	11.89
RC202-100	2,190.48	2,251.39	1,952.98	12.16	15.27
RC203-100	1,931.69	2,022.9	1,898.12	1.76	6.57
RC204-100	1,750.12	1,827.48	1,890.14	-7.4	-3.31
RC205-100	2,226.16	2,274.91	2,118.65	5.07	7.37
RC206-100	2,054.41	2,084.5	2,189.24	-6.15	-4.78
RC207-100	2,012.44	1,836.63	1,735	15.97	5.84
Moyenne				**5.31**	**5.55**
Moyenne RC1.RC2				**9.08**	**5.28**

TABLE 2.16 – Comparaison de 100 clients (C1, C2, R1, R2, RC1, RC2) pour l'approche QIHSVPS

QIHSVPS a amélioré les résultats d'ACO et PSO avec une moyenne de 9.08%, 5.28% respectivement.

Pour résumer :

- L'approche proposée EBBO améliore ACO dans 31 sur 55 instances $(56,36\%)$, et améliore PSO dans 28 instance sur $55(50,90\%)$.
- L'approche DBCS améliore ACO dans 32 sur 55 instances (58.18%), et améliore PSO dans 21 instance sur 55(54.54%).
- L'approche EDHS améliore ACO dans 23 sur 55 instances $(41,81\%)$, et améliore PSO dans 21 instance sur 55(38.18%).
- L'approche QIHSVPS améliore ACO dans 40 sur 55 instances (72.72%), et améliore PSO dans 36 instance sur 55(65.45%).

2.7.4 Analyse statistique

Puisque les résultats de chaque instance sont mutuellement indépendants,nous avons utilisé le test de Friedman [Friedman 1937] pour détecter les différences dans la performance des six approches (les quatre approches proposées ainsi qu' ACO et PSO [Belmecheri 2013]). Les résultats sont représentés dans le tableau 2.17. L'hypothèse nulle suppose que pour chaque instance, le classement des approches est tout probable.

Q (Valeur observée)	20.926
Q (Valeur critique)	11.070
Alpha	0.05

TABLE 2.17 – Résultats de test de Friedman

Au seuil de signification $\alpha = 0.05$, on peut rejeter l'hypothèse nulle d'absence de différence entre les six approches. Autrement dit, la différence entre les échantillons est significative.

Nous pouvons conclure qu'il existe au moins une métaheuristique dont la performance est différente d'au moins une des autres métaheuristiques. Pour savoir quelles métaheuristiques sont vraiment différentes, il est nécessaire d'effectuer un test de comparaisons multiples. La matrice des comparaisons par paires est présentée dans les tableaux 2.18, 2.19. La valeur critique pour la différence est 57.593. Il est à noter que dans le tableau 2.19 **NS** signifie différence non significative, et **X** signifie différence significative.

Nous remarquons dans les tableaux 2.18 et 2.19 qui résultent le test de Bonferroni-Dunn que la différence entre les résultats n'est significative que pour les approches (EDHS-QIHSVPS) et (ACO- QIHSVPS). Ceci prouve que

	EBBO	DBCS	EDHS	QIHSVPS	ACO	PSO
EBBO						
DBCS	9.500					
EDHS	54.000	44.500				
QIHSVPS	31.000	40.500	85.000			
ACO	28.000	18.500	26.000	59.000		
PSO	14.500	5.000	39.500	45.500	13.500	0.000

TABLE 2.18 – Matrice des comparaisons par paires (différence)

	EBBO	DBCS	EDHS	QIHSVPS	ACO	PSO
EBBO						
DBCS	NS					
EDHS	NS	NS				
QIHSVPS	NS	NS	X			
ACO	NS	NS	NS	X		
PSO	NS	NS	NS	NS	NS	NS

TABLE 2.19 – Matrice des comparaisons par paires (conclusion)

l'hybridation avec les principes quantiques permet d'améliorer la qualité de la solution. Pour les autres approches proposées, nous avons vu qu'il y a une amélioration mais qui est non significative pour ce test.

2.8 Conclusion

Dans ce chapitre, nous avon décrit la technique d'adaptation concernant l'ajustement des paramètres de la métaheuristique. Nous avons présenté aussi la formulation mathématique du premier problème traité (HVRPMBTW). Nous avons présenté par la suite les quatre approches proposées pour la résolution de ce problème, à savoir, l'approche inspirée de la biogéographie, l'approche inspirée du comportements des coucous, l'approche inspirée de la musique et l'approche qui hybride les principes quantiques avec l'approche inspirée de la musique.
Enfin, nous avons présenté une étude comparative des différents résultats validée par une analyse statistique. Nous allons présenter dans le chapitre suivant le deuxième problème traité (VRPSPD) ainsi que l'approche coopérative proposée.

Chapitre 3

Approches coopératives pour la résolution du problème VRPSPD

Sommaire

Le problème du tournées de véhicules avec collecte et livraison VRPPD est une extension du problème de tournées de véhicules, où les véhicules sont non seulement censés livrer des marchandises aux clients à partir d'un dépôt, mais aussi d'en collecter à partir de certains clients et de les retourner au dépôt.

Pratiquement, le VRPPD s'inscrit dans le domaine de la logistique inverse, un secteur de recherche ayant une importance croissante. Un domaine d'application très commun est le service postal. Les lettres et les colis peuvent être livrés et collectés sur la même tournée. D'un point de vue mathématique, le VRPPD est une généralisation de VRP classique donc est un problème

d'optimisation combinatoire NP-difficile. Il peut être formulé comme un ILP mixte ([Nagy 2005]) et résolu par des méthodes exactes pour des petites instances.
Le problème de tournées de véhicules avec collecte et livraison simultanées (VRPSPD) a été introduit vers la fin des années 1980 ([Min 1989]) pour résoudre un problème de distribution des livres entre 22 bibliothèques publiques (clients) et un centre d'administration de la bibliothèque (dépôt) avec un nombre fixe de véhicules de service homogènes de capacités limitées. Le VRPSPD est considéré comme un problème statique car les coûts de voyage sont déterministes dès le début.
Après son introduction en 1989 par Min [Min 1989], le problème VRPSPD a été absent des travaux de recherche pendant environ une décennie entière. En 2001 Dethloff [Dethloff 2001] a proposé une formulation précise du problème et a développé des heuristiques d'insertion à l'aide de quatre critères différents pour résoudre le problème. En 2007, Ping Chen et al. [Chen 2007] a proposé une heuristique pour résoudre le problème VRPSPD basée sur le Système des colonies de fourmis (ACS). Dans leur algorithme, la phase de construction d'ACS classique est remplacée par une procédure d'insertion alternative. Une nouvelle métaheuristiques hybride est proposée dans [Meng 2008]. Tout d'abord, l'algorithme de tour-partitionnement révisé est utilisé pour obtenir la solution initiale du VRPSPD. Deuxièmement, les modules de recherche tabou réactive ont été proposés pour améliorer les résultats. Dans [Zhang 2009b], Tao Zhang et al. ont présenté un algorithme mixte appelé PSO-ACS pour résoudre VRPSPD combinant ACS et PSO. L'algorithme proposé (PSO-ACO) améliore la vitesse de convergence en modifiant le poids d'inertie du PSO.

Le VRPSPD a été également traité dans [Liu 2010]. Des heuristiques hybrides ont été proposées pour le résoudre. Dans la même année, Goksal et al.[Goksal 2010] ont présenté une approche de résolution basée sur PSO, dans laquelle une recherche locale est effectuée par l'algorithme de descente à voisinage variable VND. Dans [Tasan 2010], Tasan et al. ont proposé une approche basée sur l'algorithme génétique pour résoudre le problème VRPSPD. Plus récemment, Subramanian et al. [Subramanian 2013b] ont proposé une approche de branch-Cut and Price BCP pour résoudre le problème VRPSPD. L'algorithme BCP a été testé sur des instances tirées de benchmarks bien connus impliquant jusqu'à 200 clients. Quatre instances ont été résolus pour la première fois et certains bornes inférieures ont été améliorées. Des solutions exactes pour le problème VRPSPD symétrique et asymétrique sont présentées dans [Rieck 2013]. Dans ce travail, deux formulations linéaires de modèles mixtes en nombres entiers sont présentées. Des instances symétriques tirées

des benchmarks connus de la littérature ainsi que de nouvelles instances asymétriques provenant de problèmes du monde réel sont résolues à l'optimalité en utilisant CPLEX. Dans [Wang 2013], les auteurs traitent une variante de VRPSPD appelé VRPSPD avec Time Windows. Pour le résoudre, ils développent un algorithme de recuit simulé (SA). Les résultats numériques sont rapportés pour un jeu de 15 instances prises des benchmarks de Wang Chen et comparés avec les résultats d'un algorithme génétique. Ils démontrent que SA est une métaheuristique efficace pour résoudre ce problème. L'algorithme d'évolution différentielle est utilisé pour optimiser les itinéraires de Anbessa City Bus Service Entreprise (ACBSE), Addis-Abeba, Ethiopie [Berhan 2014], qui est un VRP stochastique avec collecte et livraison simultanées d'un cas réel. Nous proposons dans ce chapitre de résoudre le problème VRPSPD par une approche coopérative [Berghida 2014c].

3.1 Formulation mathématique du deuxième problème

Montané et al.[Montané 2006]a formulé la variante VRPSPD comme suit :
On note :

V	l'ensemble de clients avec V_0 le dépôt des véhicules.
n	le nombre de client $n = \|V\|$.
c_{ij}	la distance entre le client i et j.
p_j	la demande de collecte du client j, $j = 1.....n$.
d_j	la demande de livraison du client j, $j = 1.....n$.
Q	Capacité d'un véhicule.
K	le nombre maximum de véhicules.
y_{ij}	la demande ramassée dans les clients acheminés jusqu'au noeud i (y compris noeud i) et transportés sur l'arc (i, j).
z_{ij}	la demande livrée aux clients en route après le client i et transportée sur l'arc (i, j).

La variable de décision x_{ij} où :

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

La fonction objectif du VRPSPD :

$$\text{Min} \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij}.x_{ijk} \tag{3.1}$$

Sous les contraintes suivantes :

$$\sum_{i=0}^{n}\sum_{k=1}^{K} x_{ijk} = 1; j = 1...n \tag{3.2}$$

$$\sum_{i=0}^{n} x_{ijk} - \sum_{i=0}^{n} x_{jik} = 0; j = 1...n, k = 0...K \tag{3.3}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1; k = 0...K \tag{3.4}$$

$$\sum_{i=0}^{n} y_{ji} - \sum_{i=0}^{n} y_{ij} = p_j, \forall j \neq 0 \tag{3.5}$$

$$\sum_{i=0}^{n} z_{ij} - \sum_{i=0}^{n} z_{ji} = d_j, \forall j \neq 0 \tag{3.6}$$

$$y_{ij} + z_{ij} \leq Q \sum_{k=1}^{K} x_{ijk}; \forall i, j = 0...n \tag{3.7}$$

$$x_{ij} \in [0, 1]; i, j = 0...n \tag{3.8}$$

$$y_{ij} \geq 0; \forall i, j = 0...n \tag{3.9}$$

$$z_{ij} \geq 0; \forall i, j = 0...n \tag{3.10}$$

La fonction objectif (3.1) montre que ce modèle minimise le coût de calcul d'itinéraire. La contrainte (3.2) assure qu'un client est visité exactement par un véhicule. La contrainte (3.3) garantit que le même véhicule arrive et repart de chaque client servi. (3.5) assure que la plupart des véhicules sont utilisés. (3.6) et (3.7) sont des équations pour la collecte et les demandes de livraison, respectivement, ils garantissent que les contraintes soient satisfaites pour chaque client. (3.8) implique que les collectes et livraisons seront transportées en utilisant des arcs inclus dans la solution de plus, elle impose la limite supérieure de la charge d'un véhicule. Enfin, (3.9),(3.10) et (3.11) définissent la nature des variables de décision.

3.2 Technique de coopération proposée

Pour résoudre un problème d'optimisation, une méthode qui paraît bien adaptée est choisie parmi celles existantes (méthode exacte, heuristiques,...etc.). Ensuite, des essais d'amélioration sont apportés en travaillant sur ses paramètres afin d'obtenir la méthode la plus efficace possible. Si c'est

possible, sa qualité est évaluée en la comparant à d'autres méthodes proposées pour le problème étudié. Malheureusement, d'après les *No Free Lunch Theorems* [Wolpert 1997], il n'existe pas de métaheuristique qui soit meilleure que toutes les autres métaheuristiques pour tous les problèmes. Dans la pratique, il existera toujours des instances pour lesquelles une métaheuristique est meilleure qu'une autre. Quelque soit la métaheuristique choisie, elle présente des avantages et des inconvénients.
L'idée de coopérer les algorithmes de recherche est classique en optimisation combinatoire. Dans la co-évolution coopérative CC [coo 2014], les collaborations entre l'ensemble d'individus sont nécessaires afin d'évaluer une solution complète. CC a été présenté comme une approche évolutive alternative pour optimiser les fonctions [Potter 1994]. Pour cela, ils considèrent que le nombre de variables d'une fonction est égal au nombre de populations et chaque variable est un composant de la solution qui est traitée séparément en utilisant un algorithme évolutionnaire. Pour résoudre un problème en utilisant CC, tout d'abord, nous devons décomposer le problème en sous-problèmes. Chaque sous-problème est attribué à une population. Toutes les populations évoluent au même temps. Elles n'échangent les informations que dans l'étape d'évaluation des solutions (calcul de fitness). Chaque individu dans une population représente une partie de la solution et une solution candidate potentielle pour chaque composant (sous-problème). Il ne peut pas être évalué séparément des autres parties complémentaires.
La coopération dans le domaine des métaheuristiques pourrait également prendre la forme d'algorithmes hybrides qui sont connus pour être supérieurs aux algorithmes de recherche pures concernant la qualité des optima trouvés et le temps pris pour les trouver [Preux 1999] [Fleurent 1994]. L'hybridation entre algorithmes prend généralement l'une des formes suivantes :

- **Hybride séquentiel :** lorsque deux algorithmes sont appliqués l'un après l'autre ; les résultats fournis par le premier, le deuxième solutions initiales à la manière d'un pipeline ;
- **Hybride parallèle synchrone :** où un algorithme est utilisé à la place d'un opérateur ;
- **Asynchrone hybride parallèle :** où plusieurs algorithmes de recherche travaillent concurremment et échangent des informations.

Les hyper-heuristiques peuvent être définies comme des méthodes d'optimisation ayant la caractéristique d'utiliser une approche heuristique pour sélectionner les heuristiques afin de résoudre un problème d'optimisation. L'approche hyper-heuristique propose de regrouper un ensemble d'heuristiques ou métaheuristiques et d'établir un mécanisme pour identifier et sélectionner les méthodes de recherche les plus efficaces au cours du processus d'optimisation. Généralement, certaines études visent à produire

des heuristiques constructives qui construiront une solution, étape par étape. Les heuristiques sont utilisées pour décider comment étendre une solution partielle. Ces méthodes ont tendance à être rapide. D'autres études visent à produire des heuristiques d'amélioration ou de perturbation travaillant sur une solution candidate déterminée et essayant d'améliorer sa qualité. Ces méthodes sont plus lentes mais fournissent les meilleurs résultats finaux. Certaines études sont des hybridations entre les deux méthodes.

Nous proposons dans cette thèse une autre forme de coopération entre les métaheuristiques où une métaheuristique maitre contrôle plusieurs métaheuristiques esclaves. Les points forts d'une métaheuristique esclave compensent les points faibles d'une autre grâce à la notion de **rang**. Le rang est un moyen de mettre en valeur les points forts des métaheuristiques esclaves. Il est mis à jour continument par la métaheuristique maitre qui augmente le rang des heuristiques ayant amélioré la solution et diminue le rang de celles qui ne l'auront pas amélioré. Cette mise à jour tend à récompenser les heuristiques esclaves qui auront réussi à améliorer la solution courante. Le rang aide l'heuristique maitre à choisir la meilleure heuristique esclave à chaque point de décision en choisissant l'heuristique esclave ayant le plus grand rang. L'algorithme 11 présente le processus de coopération, où :

- δ représente la différence de bénéfice entre la solution trouvée par l'une des métaheuristiques esclaves et la solution courante.
- α représente la différence entre la qualité de la solution trouvée et celle de la solution initiale. Si la qualité de la solution est améliorée, α sera positif, sinon α sera négatif

Algorithme 11 Algorithme de coopération

tant que condition d'arret non vérifiée **faire**
Choisir la métaheuristique k, avec le plus haut rang et $k \in$ Liste Taboue.

si $\Delta > 0$ **alors**
$r_k = r_k + \alpha$ et vider la Liste Taboue.
sinon
$r_k = r_k + \alpha$ et inclure k dans la Liste Taboue.

La concurrence entre les métaheuristiques esclaves favorise l'intensification de la solution dans l'espace de recherche. La coopération entre les esclaves est introduite par le mécanisme suivant : lorsqu'une métaheuristique termine sa recherche, elle communique sa meilleure solution à celle qui la remplace, permettant ainsi à cette dernière de générer un bon voisinage.
La principale différence entre l'approche coopérative proposée et CC est le

mécanisme d'activation de métaheuristiques pour chercher une solution. Dans notre cas, nous avons utilisé la notion de rang pour activer une métaheuristique, tandis que dans CC ; l'exécution dse métaheuristiques est parallèle. Dans notre cas, il n'y a pas de décomposition de problème, chaque métaheuristique esclave manipule une solution complète qui peut être utilisée par d'autres métaheuristiques. Toutefois, CC traite des sous-problèmes où chaque composant gère une partie de la solution. La différence principale entre les hyper-heuristiques et l'approche coopérative proposée que nous manipulons des solutions complètes, tandis que les hyper-heuristiques manipulent des solutions partielles.

3.3 Encodage de la solution

3.3.1 Définition de la structure de données

La structure de données adoptée est similaire à celle décrite dans le chapitre 2, dans la section 2.3.2.1.

3.3.2 Génération d'une solution

Démarrer le processus de recherche par une bonne solution initiale peut être important si des solutions de haute qualité doivent être atteintes aussi rapidement que possible. Les choix standards pour générer une solution initiale sont soit une solution initiale aléatoire ou une solution retournée par une heuristique. Dans un problème de tournées de véhicules, la façon la plus simple est d'affecter aléatoirement chaque client à un véhicule tout en respectant les contraintes du problème. Dans le problème VRPSPD, le nombre de véhicule est illimité [Montané 2006]. Pour générer une solution faisable aléatoirement, nous commençons par un véhicule, et nous affectons des clients aléatoires à ce véhicule. une fois le véhicule rempli, nous ajoutant un autre véhicule et ainsi de suite jusqu'à ce que tous les clients soient servis.
Pour vérifier la faisabilité de la solution, il suffit de simuler le parcours de chaque véhicule de la solution. Cela se fait en soustrayant de la charge du véhicule, à chaque arrêt de service (visite chez un client) la quantité de marchandise demandée par le client (livraison) et y ajoutant la quantité devant en être prise (collecte). La charge du véhicule doit être inférieure à sa capacité. La procédure de génération est décrite par l'algorithme 12.

Même si cette méthode de génération est simple et rapide, l'algorithme peut prendre plusieurs itérations avant d'atteindre une solution de qualité. Pour améliorer cette dernière, la notion de clustering est introduite. Cette approche vise à regrouper les clients, dont les coordonnées sont les plus proches.

Algorithme 12 Création d'une solution

tant que Les clients c_i ne sont pas satisfaits **faire**
  Générer un véhicule V vide $charge_V := 0$;
  tant que $charge_V <= capacity$ **faire**
    Remplir aléatoirement V avec les demandes de livraisons d_i ;
    $charge := charge + d_i$;
Simulation de parcours :
pour chaque V_i **faire**
  pour chaque client c_k de V_i **faire**
    $charge_V := charge_V + p_k - d_k$;
Valider la solution ;

Dans la génération initiale, nous tenterons d'affecter au même véhicule les clients appartenant au même groupe. Cette méthode améliore considérablement la qualité moyenne des solutions de départ générées. Le graphe présenté dans la figure 3.1 illustre pour différentes instances du problème, le gain dans la qualité des solutions obtenu par cette méthode.Nous avons utilisé la méthode K-means [MacQueen 1967] pour implémenter le clustering.

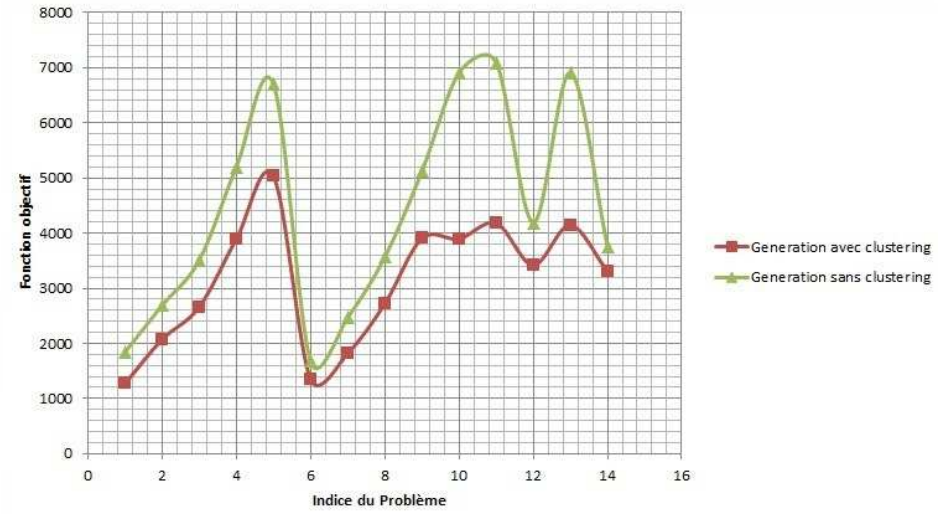

FIGURE 3.1 – Comparaison des méthodes de génération

3.3.2.1 L'algorithme des K-moyennes (K-means)

K-moyennes est un algorithme qui permettre de classer ou de regrouper les objets en fonction des attributs / caractéristiques en K groupes où K est nombre entier positif. Le regroupement se fait en minimisant la somme

Algorithme 13 Génération d'une solution avec clustering

pour chaque $Cluster_j$ **faire**
  Générer un véhicule V vide $charge_V := 0$;
  tant que Les clients c_i ne sont pas satisfaits **faire**
    tant que $charge_V <= capacity$ **faire**
      Remplir aléatoirement V avec les demandes de livraisons d_i ;
      $charge_V := charge_V + d_i$;
  Simulation de parcours :
  pour chaque V_i **faire**
    pour chaque client c_k de V_i **faire**
      $charge_V := charge_V + p_k - d_k$;
  Valider la solution ;

des carrés des distances entre les données et le centre de gravité du cluster correspondant. Ainsi, le but de regroupement de K-moyenne est de classer les données [Ombuki 2006].

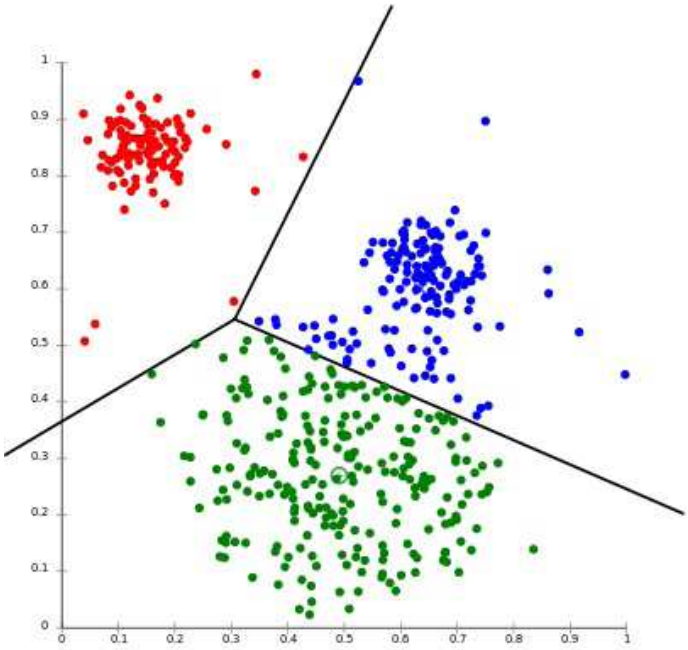

FIGURE 3.2 – Groupement des individus avec la méthode K-moyennes

Les étapes de base de K-means sont simples. Au début, nous déterminons

un nombre K de clusters et nous supposons le centre de gravité ou le centre de ces groupes. Nous pouvons prendre des objets aléatoires ou bien les K premiers objets comme centres de gravité initiaux. Ensuite, le K-means fera les trois étapes ci-dessous jusqu'à convergence :
Itérer jusqu'à stabilité (aucun objet ne se déplace de son groupe) :

1. Déterminer le centre de gravité de coordonnées
2. Déterminer la distance de chaque objet aux centres de gravité
3. Grouper les objets en fonction de la distance minimale (trouver le centre de gravité le plus proche)

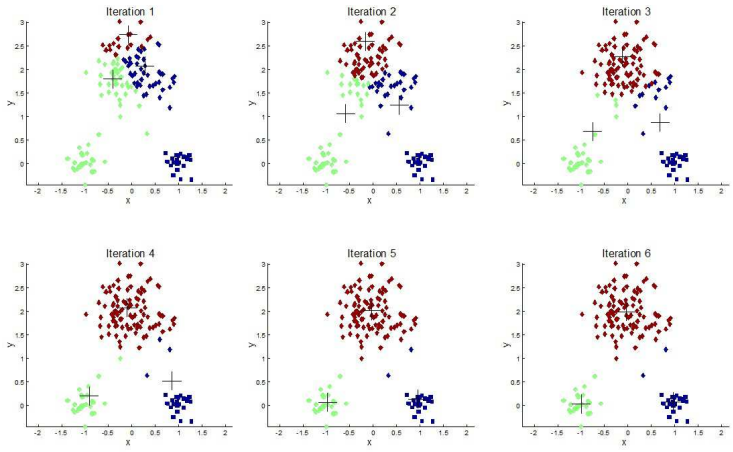

FIGURE 3.3 – Exemple de déroulement de l'algorithme K-means

L'algorithme K-means est assez rapide, ce qui est bénéfique pour la génération aléatoire initiale.

3.3.3 Rectification d'une solution

La génération des solutions initiales par K-means ne garantit pas la faisabilité de la solution en terme de capacité. En effet, la méthode regroupe les clients par rapport à la distance. Cela ne peut pas garantir que les contraintes de capacité ne soient pas violées. Rien n'assure que les clients proches en terme de distance, auront une demande globale inférieure à la capacité du véhicule. Afin d'éviter des temps exorbitants dans la génération des solutions initiales, une méthode de rectification s'avère indispensable. Cette procédure consiste à éliminer de la tournée les clients ayant causé la non faisabilité de la solution, pour les affecter à d'éventuelles autres tournées sans toucher à leurs faisabilité. Les clients ne pouvant pas être affectés, constitueront de nouvelles tournées. La procédure de rectification est illustré par l'algorithme 14.

Algorithme 14 Rectification d'une solution

Entrée : solution S ;
$i := 0$;
tant que $i < nombreVehicule_S$ **faire**
  si $V_{S_i} > capacity$ **alors**
    Sauvegarder(i) dans liste l ;
    tant que $surpoids(V_{Si})$ **faire**
      Supprimer aléatoirement un client C_j ;
  $i := i + 1$;
pour chaque C_j supprimé **faire**
  Distribution aléatoire sur les véhicules $V_{S_{k \notin l}}$;
Vérifier solution S ;
Retourner solution S ;

3.3.4 Méthodes de génération du voisinage

Le calcul du voisinage d'une solution a un impact direct sur la qualité de cette dernière. Pour cela, nous proposons trois types de calcul de voisinage et essayons de dégager le mécanisme qui offre le plus d'avantage. Nous présentons dans ce qui suit les trois types de voisinages.

1. **Calcul de voisinage par permutation de clients d'une même tournée :** Dans cette méthode, les voisins sont calculés par permutation des clients au sein d'une même tournée (Algorithme 15), ceci revient à modifier l'ordre de visite des clients par le véhicule. La figure 3.4 illustre ce principe de calcul de voisinage.

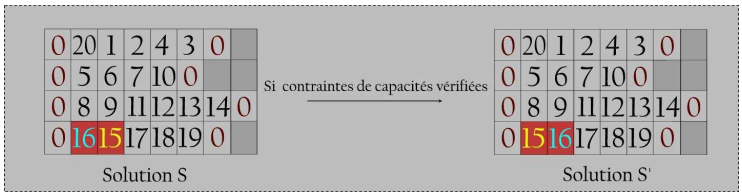

FIGURE 3.4 – Illustration du mouvement de permutation dans une même tournée

2. **Calcul de voisinage par permutation de clients entre deux tournées :** Cette méthode de calcul consiste à permuter deux clients choisis aléatoirement entre deux tournées différentes tout en vérifiant la contrainte de capacité (Algorithme 16). La figure 3.5 illustre le mécanisme de cette permutation.
3. **Calcul de voisinage par déplacement d'un client :** Cette méthode de calcul consiste à choisir aléatoirement un client d'une tournée

Algorithme 15 Mouvement de permutation de clients dans une même tournée

Choisir aléatoirement un véhicule V_i d'une solution S ;
Choisir aléatoirement deux clients c_j,c_k dans V_i ;
Permuter c_j et c_k ;
si verfierSolution(S)=vrai **alors**
  Retourner vrai ;
Retourner faux ;

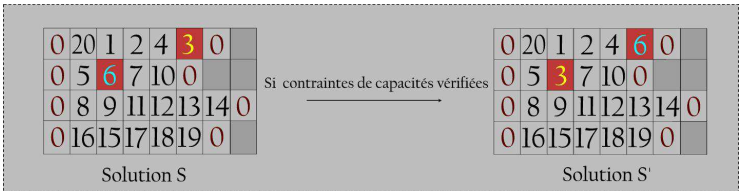

FIGURE 3.5 – Permutation entre deux tournées

Algorithme 16 Mouvement de permutation de clients entre deux tournées

Choisir aléatoirement deux véhicules V_i et V_j d'une solution S et S^* ;
Choisir aléatoirement deux clients c_k,c_l dans V_i et V_j ;
Permuter c_k et c_l ;
si $verfierSolution(S)$ et $verfierSolution(S^*)$ **alors**
  Retourner vrai ;
Retourner faux ;

et à l'affecter à une autre tournée tout en respectant la contrainte de capacité (Algorithme 17). La figure 3.6 illustre le mécanisme de cette permutation.

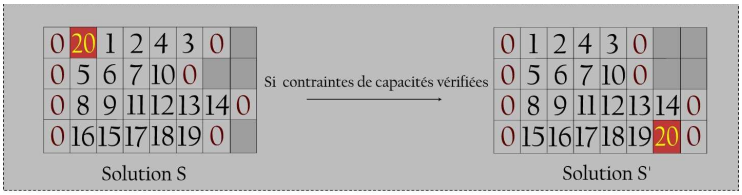

FIGURE 3.6 – Mouvement de déplacement d'un client d'une tournée à une autre

Algorithme 17 Mouvement de déplacement d'un client (d'une tournée à une autre)

Choisir aléatoirement deux Véhicules V_i et V_j d'une Solution S ;
Choisir aléatoirement un client $c$dans V_i ;
Déplacer c vers V_j ;
si verfierSolution(S_i) **alors**
  Retourner vrai ;
Retourner faux ;

3.4 Métaheuristiques utilisées dans la coopération

L'approche de coopération utilisée, utilise la métaheuristique Tabu Search comme maitre et les trois métaheuristiques Recuit simulé, l'algorithme génétique et l'optimisation basée sur la biogéographie comme esclaves. La métaheuristique maitre travaille dans l'espace des métaheuristiques, et les métaheuristiques esclaves dans l'espace des solutions. Nous allons passer en revue chacune de ces métaheuristiques.

3.4.1 Première métaheuristique utilisée : Le recuit simulé

3.4.1.1 Le principe du recuit simulé

Le Recuit Simulé (Simulated Annealing SA) est inspiré du processus de recuit dans la métallurgie. Dans ce procédé, un matériau naturel est chauffé et refroidi lentement dans des conditions contrôlées pour augmenter la taille des cristaux dans le matériau et de réduire les défauts. Ceci améliore la résistance et la durabilité du matériau. La chaleur augmente l'énergie des atomes, leur permettant de se déplacer librement, et le programme de refroidissement lent permet qu'une nouvelle configuration à faible énergie à être découverte et exploitée. C'est l'idée clé qui sous-tend l'algorithme du recuit simulé, qui a été proposé indépendamment par Kirkpatrick et al. [Kirkpatrick 1983b] et Cern'y [Černý 1985b]. La méthode est une adaptation de l'algorithme de Metropolis-Hastings, une méthode de Monte Carlo pour générer des états exemples d'un système thermodynamique, inventée par Rosenbluth et publiée dans un article de Metropolis et al.[Metropolis 1953].
Chaque configuration d'une solution dans l'espace de recherche constitue une énergie interne différente du système. Le chauffage des systèmes conduit en un assouplissement des critères d'acceptation des échantillons prélevés dans

l'espace de recherche. Comme on refroidit le système, les critères d'acceptation des échantillons se rétrécissent pour se concentrer sur l'amélioration de mouvements. Une fois le système refroidi, la configuration représentera un échantillon qui a ou est près d'un optimum global. [Brownlee 2011].
L'idée est de permettre des modifications locales susceptibles d'augmenter le coût (si cette fonction est à minimiser). Ces modifications vont peut-être permettre de franchir une barrière et aboutir à une solution meilleure.
Le processus de recuit simulé est contrôlé par différents paramètres. Les bonnes valeurs pour ces paramètres ne sont pas précisément connues. Le processus doit donc être évalué avant d'être utilisé à l'échelle réelle de façon à ajuster ces divers paramètres. Supposons que le coût d'une configuration X_i soit $f(X_i)$. Une modification locale fait passer à une configuration X_j de coût $f(X_j)$. la décision d'accepter X_j à la place de X_i est prise de la manière suivante :
Soit $\Delta f_{ij} = f(X_j) - f(X_i)$;

- si $\Delta f_{ij} \leq 0$, accepter X_j ;
- si $\Delta f_{ij} > 0$ accepter X_j avec la probabilité $exp(-\Delta f_{ij}/T)$, ou T est appelée température.

Cette distribution de probabilité est inspirée de la loi de physique statistique de Gibbs-Boltzmann. Dans le contexte de l'informatique, ce choix est connu sous le nom de règle de Metropolis.
On voit que, plus la température est élevée, plus la probabilité d'accepter une configuration moins bonne est grande. Par contre, à la fin du processus(lorsque la température est basse), seules les modifications faisant diminuer le coût seront acceptées.

L'algorithme 18 présente le pseudo code de l'algorithme du recuit simulé pour un problème de minimisation.

Algorithme 18 Algorithme de Recuit Simulé

$S_{courante} = CreerSolInitiale()$
$S_{best} = S_{courante}$
pour $i = 1$ à $Nbr_Iteration$ **faire**
  $S_i = CreerSolVoisine(S_{courante})$
  $Temperature_{courante} = CalculerTemp(i, Temperature_{max})$
  si $Cout(S_i \leq Cout(S_{courante}))$ **alors**
    $S_{courante} = S_i$
    si $Cout(S_i) \leq Cout(S_{best}))$ **alors**
      $S_{best} = S_i$
  sinon
    si $exp(\frac{Cout(S_{courante}) - Cout(S_i)}{Cout(S_{courante})}) > Rand()$ **alors**
      $S_{courante} = S_i$

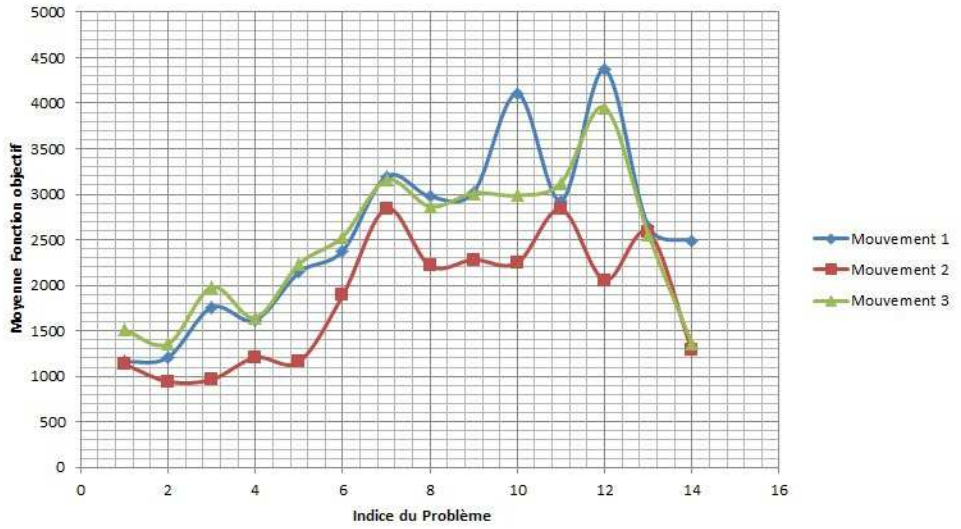

FIGURE 3.7 – Graphe des différents mouvements sur l'ensemble des instances

3.4.1.2 Adaptation du recuit simulé pour résoudre le problème VRPSPD

Dans cette section, nous allons passer en revue les différentes adaptations faites pour résoudre le problème VRPSPD. La solution initiale est générée à l'aide d'un clustering des clients en utilisant K-means suivi d'une rectification permettant d'engendrer une solution faisable (Algorithme 13).

L'initialisation de la température est un des paramètre décisif du nombre d'itérations de l'algorithme. L'évolution de la température est une fonction réalisant une diminution graduelle de la température. La modification de la température se fait par décrémentation après un certain nombre d'itérations, ainsi la température resterai inchangée pendant le nombre d'itération représentant un palier. La condition d'arrêt du palier de l'algorithme du recuit simulé est exprimée en terme de taux d'acceptation, c'est à dire qu'à une même température, nous continuons à explorer l'espace de recherche tant que ce taux est inférieur à un certain seuil.

Afin de déterminer la meilleure façon de générer la solution voisine, nous avons mené une étude comparative entre les trois modes de calcul du voisinage introduits dans la section 5.3.4 (Figure 3.7). Nous avons opté pour le calcul de

voisinage par permutation de clients entre deux tournées. Nous avons constaté qu'à partir d'une certaine qualité de la solution (point critique), ce calcul de voisinage perd son efficacité. Pour cela, nous avons adopté un autre type de calcul de voisinage à partir de ce point critique. Il s'agit du calcul de voisinage par permutation de deux clients au sein d'une même tournée. Ce changement dans le calcul du voisinage va nous permettre une plus grande exploration de l'espace de recherche.
La figure 3.8 montre clairement l'amélioration introduite par le changement du mode de calcul du voisinage pendant le processus du recuit simulé.

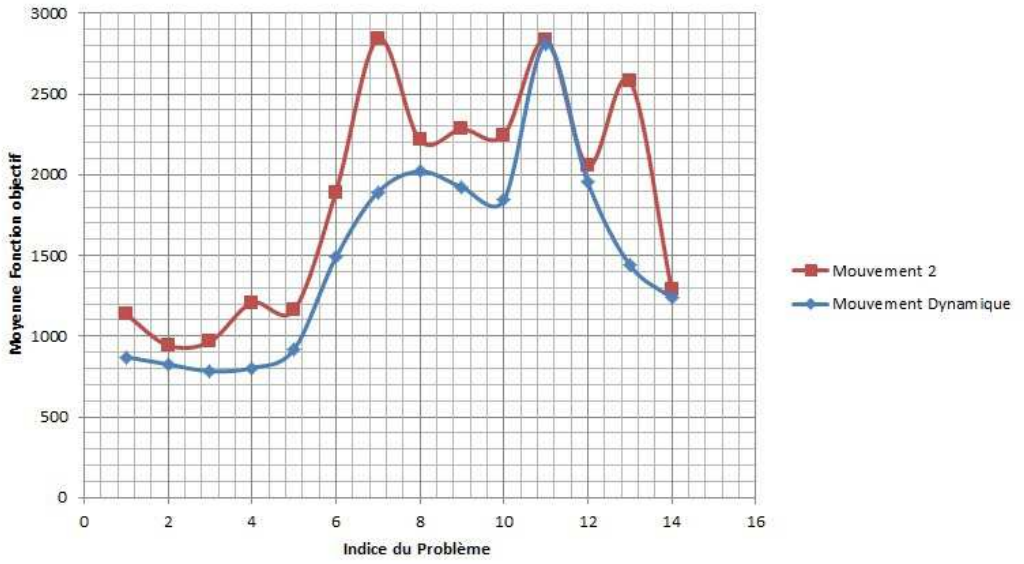

FIGURE 3.8 – Graphe comparatif du mouvement 2 avec le mouvement dynamique sur l'ensemble des instances

3.4.2 Deuxième métaheuristique utilisée : L'algorithme génétique

3.4.2.1 Principe général de l'algorithme génétique

Les algorithmes génétiques ont déjà une histoire relativement ancienne puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent à 1962 [Holland 1962]. L'ouvrage de David Goldberg [Goldberg 1989] a largement contribué à les vulgariser. Les algorithmes génétiques (AGs) sont des algorithmes de recherche d'usage général basés sur les principes de l'évolution observée dans la nature. Les algorithmes génétiques combinent la sé-

lection, le croisement, et les opérateurs de mutation dans le but de trouver la meilleure solution d'un problème. Ils cherchent la solution optimale jusqu'à ce qu'un critère de terminaison spécifié soit atteint. La solution du problème est appelée chromosome. Un chromosome est constitué d'une collection de gènes qui sont tout simplement les paramètres à optimiser. Un algorithme génétique crée une population initiale (une collection de chromosomes), évalue cette population, puis évolue la population à travers plusieurs générations (en utilisant les opérateurs génétiques discutés ci- dessus) dans la recherche d'une bonne solution pour un problème donné.
Les AGs peuvent être appliqués à un large éventail de problèmes d'optimisation tels que la planification, les jeux informatiques, opérations boursières, le transport, le problème des voyageurs de commerce, etc.

Les phases d'un algorithme génétique

Génération de la population initiale La population initiale est générée de manière aléatoire. Il est toujours intéressant de répartir les individus de cette population sur l'espace de recherche, de sorte que cet espace soit parcouru au maximum.

Évaluation de la qualité d'une solution Chaque chromosome apporte une solution potentielle au problème à résoudre. Néanmoins, ces solutions n'ont pas toutes le même degré de pertinence. C'est à la fonction de performance (fitness) de mesurer cette efficacité pour permettre à l'algorithme de faire évoluer la population dans un sens bénéfique pour la recherche de la meilleure solution. Autrement dit, la fonction de performance f, doit pouvoir attribuer à chaque individu un indicateur positif représentant sa pertinence pour le problème qu'on cherche à résoudre.

La sélection Cet opérateur détermine la capacité de chaque individu à persister dans la population et à se diffuser. En règle générale, la probabilité de survie d'un individu sera directement reliée à sa performance relative au sein de la population. Cela traduit bien l'idée de la sélection naturelle : les gènes les plus performants ont tendance à se diffuser dans la population tandis que ceux qui ont une performance relative plus faible ont tendance à disparaître. Il existe différents types de sélection, nous pouvons citer à titre d'exemple :

- **Sélection par roulette (wheel)** Les parents sont sélectionnés en fonction de leur performance. Meilleur est le résultat codé par un chromosome, plus grandes sont ses chances d'être sélectionné. Il faut imaginer

une sorte de roulette de casino sur laquelle sont placés tous les chromosomes de la population, la place accordée à chacun des chromosomes étant en relation avec sa valeur d'adaptation. Cette roulette est représentée par la figure 3.9. Ensuite, la bille est lancée et s'arrête sur un

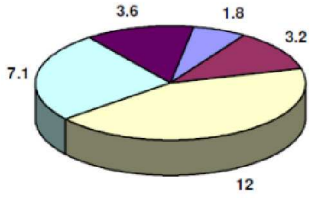

FIGURE 3.9 – Exemple de sélection par roulette

chromosome. Les meilleurs chromosomes peuvent ainsi être tirés plusieurs fois et les plus mauvais ne jamais être sélectionnés. Cela peut être simulé par l'algorithme suivant :

1. On calcule la somme $S1$ de toutes les fonctions d'évaluation d'une population.
2. On génère un nombre r entre 0 et $S1$.
3. On calcule ensuite une somme $S2$ des évaluations en s'arrêtant dès que r est dépassé.
4. Le dernier chromosome dont la fonction d'évaluation vient d'être ajoutée est sélectionné.

– **Sélection par rang**

La sélection précédente rencontre des problèmes lorsque la valeur d'adaptation des chromosomes varient énormément. Si la meilleure fonction d'évaluation d'un chromosome représente 90% de la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'évolution.
La sélection par rang trie d'abord la population par fitness. Ensuite, chaque chromosome se voit associé un rang en fonction de sa position. Ainsi le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N (pour une population de N chromosomes). La sélection par rang d'un chromosome est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais.

- **Sélection steady-state** L'idée principale est qu'une grande partie de la population puisse survivre à la prochaine génération. L'algorithme génétique fonctionne alors de la manière suivante : A chaque génération, quelques chromosomes sont sélectionnés (parmi ceux qui ont le meilleur coût) pour créer des chromosomes fils. Ensuite les chromosomes les plus mauvais sont retirés et remplacés par les nouveaux. Le reste de la population survie à la nouvelle génération.
- **Sélection par tournoi** Sur une population de m chromosomes, on forme m paires de chromosomes. Dans les paramètres de l'AG, on détermine une probabilité de victoire du plus fort. Cette probabilité représente la chance qu'a le meilleur chromosome de chaque paire d'être sélectionné. Cette probabilité doit être grande (entre 70% et 100%). A partir des m paires, on détermine ainsi m individus pour la reproduction.
- **Élitisme** A la création d'une nouvelle population, il y a de grandes chances que les meilleurs chromosomes soient perdus après les opérations de croisement et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleurs solutions.

Les opérateurs génétiques

- **Le croisement :** après avoir fait une sélection des individus il nous faut générer une nouvelle population, afin d'enrichir la diversité de la population. Le croisement permet cela en manipulant la structure des chromosomes. Il existe plusieurs manières d'effectuer un croisement soit par cross-over où l'on a besoin de deux parents qui génèrent à la fin du croisement deux enfants, soit par copie où l'on n'a besoin que d'un seul parent qui nous donnera à la fin du croisement un enfant qui est le parent lui-même (sa copie).
- **Mutation :** la mutation permet de changer (permuter) un gène d'un chromosome par un autre d'une manière aléatoire, ce qui est à première vue très ressemblant avec un cross-over. La différence est que le cross-over essaie de converger vers une solution qui lui parait la meilleure (s'intéresse à la qualité), mais que la mutation permet la diversité. En quelque sorte, la mutation sert à éviter une convergence prématurée de l'algorithme. Par exemple lors de la recherche d'une solution optimum, la mutation sert à éviter la convergence vers un optimum local. Il est nécessaire de choisir pour ce taux une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de

sélection et d'évolution.

3.4.2.2 Adaptation de l'algorithme génétique pour résoudre le problème VRPSPD

L'algorithme génétique adopté utilise les trois opérateurs conventionnels à savoir, la sélection, le croisement et la mutation. Le bon choix de la population initiale peut accélérer la convergence de l'algorithme, pour cela, nous avons opté à un clustering des clients, qui va permettre de regrouper les clients les plus proches dans la même tournée minimisant ainsi le coût des tournées dans la solution initiale.
La procédure permettant de générer la population de solutions est décrite dans l'algorithme 19

Algorithme 19 Génération population algorithme génétique

$i = 0$
tant que $i < taillePopulaiton$ **faire**
  Générer une solution S_i
  Évaluer S_i
  si $S_i \notin Population$ **alors**
    Ajouter S_i à Population
    $i = i + 1$

Il existe plusieurs opérateurs de sélection, chaque opérateur ayant ses avantages et ses inconvénients. Nous avons opté pour la roulette. Ce choix est motivé par le besoin d'équité envers toutes les solutions tout en maintenant une certaine priorité aux bonnes solutions.
Afin de vérifier cette équité , nous avons appliqué cet opérateur pour sélectionner 20 individu parmi 80. Nous remarquons sur le graphe 3.10 l'absence d'une concentration des individus choisis autour d'un point.

La procédure de sélection est décrite dans l'algorithme 20

Le croisement utilisé est un croisement en un point aléatoire. La figure 3.11 illustre un exemple de croisement de solutions. Le croisement peut produire des solutions infaisables (clients en double ou/et clients manquants). Pour les rendre faisables, nous commençons par supprimer les clients qui ont un nombre d'occurrence supérieur à un, ensuite nous ajoutons les clients manquants aux tournées. Cette correction est illustrée par la figure 3.12.

Parmi les types de mutations existants, nous trouvons la mutation consistant à permuter deux clients au sein d'une même tournée et la mutation permutant deux clients de deux tournées différentes. nous avons mené une étude comparative entre ces deux types de mutations et la mutation résultante de

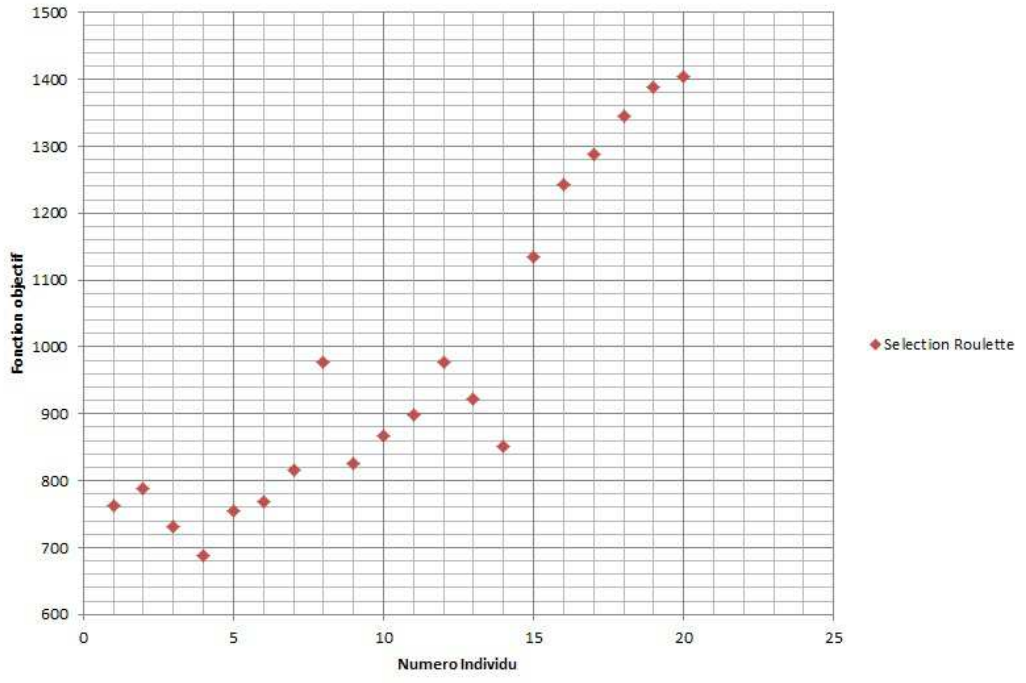

FIGURE 3.10 – sélection de 20 individus sur 80

Algorithme 20 Algorithme de sélection par roulette

$i = 0$
Trier la Population selon les qualités respectives de ses individus
Calculer la fitness totale de la Population $FITNESS$
$l = random(0; FITNESS)$
Parcourir la Population et soustraire la Fitness de l'individu courant à l jusqu'à ce que l s'annule :
$i = 0$
tant que $i < taille_{Population}$ et $l >= 0$ **faire**
  $l = l - fit_{S_i}$
  $i++$
Retourner la solution à la position l

leur hybridation. Les résultats de cette étude sont représentés dans le graphe 3.13. A l'issue de cette étude, nous avons retenu la mutation hybride qui est décrite dans l'algorithme 22.

L'algorithme global d'adaptation de l'algorithme AG pour résoudre le VRPSPD est présenté dans l'algorithme 23.

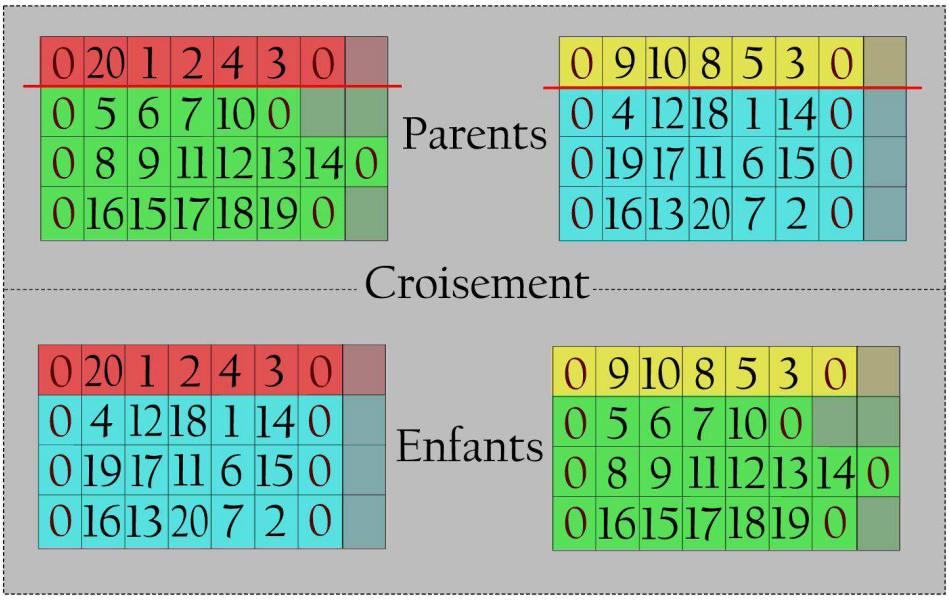

FIGURE 3.11 – Opérateur de croisement, Algorithme génétique

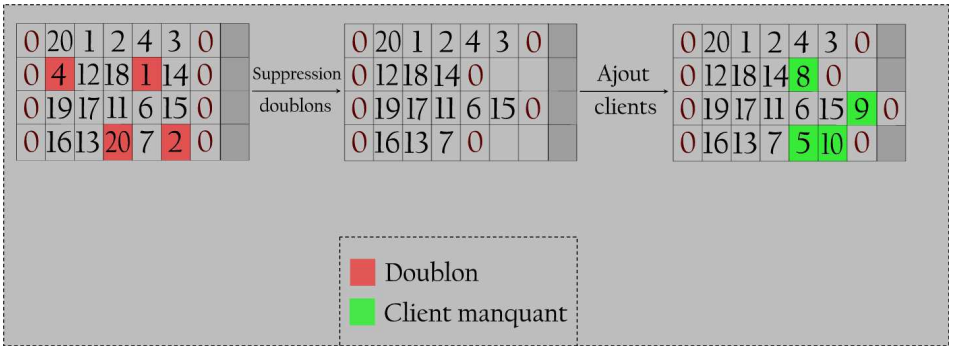

FIGURE 3.12 – Illustration de la correction d'une solution enfant.

3.4.3 Troisième métaheuristique : L'optimisation basée sur la biogéographie

3.4.3.1 Adaptation de l'optimisation basée sur la biogéographie pour résoudre le VRPSPD

Le processus basé sur la biogéographie commence par la génération d'une population initiale d'une manière similaire à celle des algorithmes génétiques cité ci-dessus. L'affectation du nombre d'espèces pour chaque solution s'effectue en triant l'ensemble des solutions dans l'ordre croissant des fitness et en affectant le rang de la solution à ce nombre d'espèces. Ainsi, dans une population de n solutions, la meilleure solution aura pour nombre d'espèces la valeur n, et la plus mauvaise aura la valeur 1. Le nombre d'espèces intervient dans

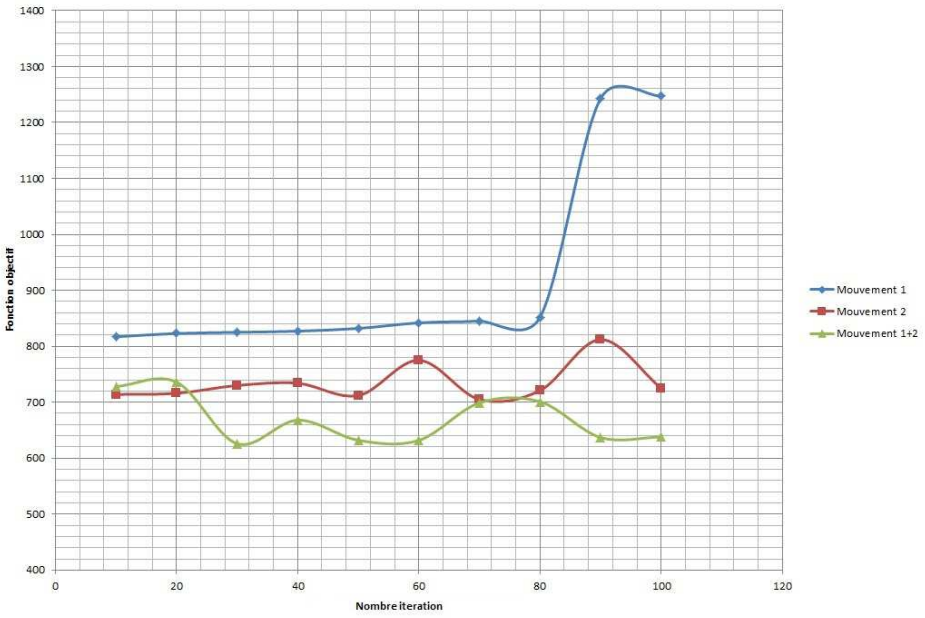

FIGURE 3.13 – Graphe mutation algorithme génétique sur 100 itérations

le calcul des taux de migration.

Le processus de migration consiste à partager les caractéristiques des bonnes solutions avec celles des mauvaises. Deux types de migration sont envisageables, la première consiste à faire migrer un certain nombre de clients d'une tournée vers une autre tournée. La deuxième consiste à faire migrer la tournée complète. Ces deux modes de migration sont illustrés par les exemples 3.14, 3.15 respectivement.

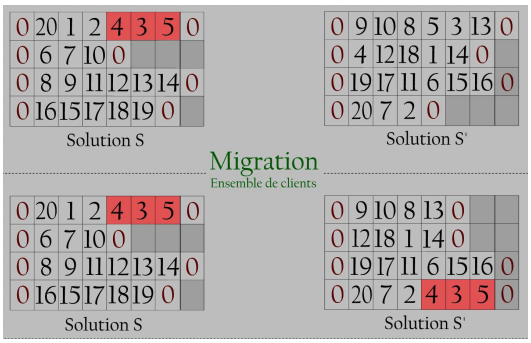

FIGURE 3.14 – Illustration migration ensemble de clients

Nous avons mené une étude expérimentale pour choisir l'un des deux modes de mutation. La figure 3.16 illustre les résultats de cette étude qui nous a mené à choisir le deuxième mode de migration.

Le processus de migration est décrit dans l'algorithme 24.

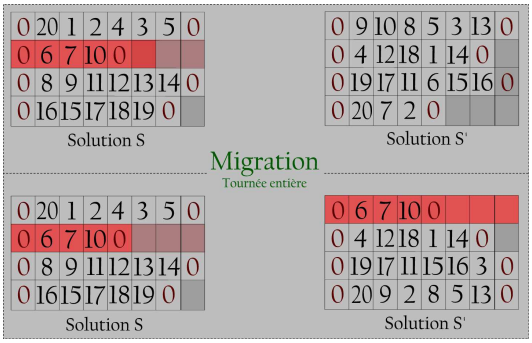

FIGURE 3.15 – Illustration migration véhicule

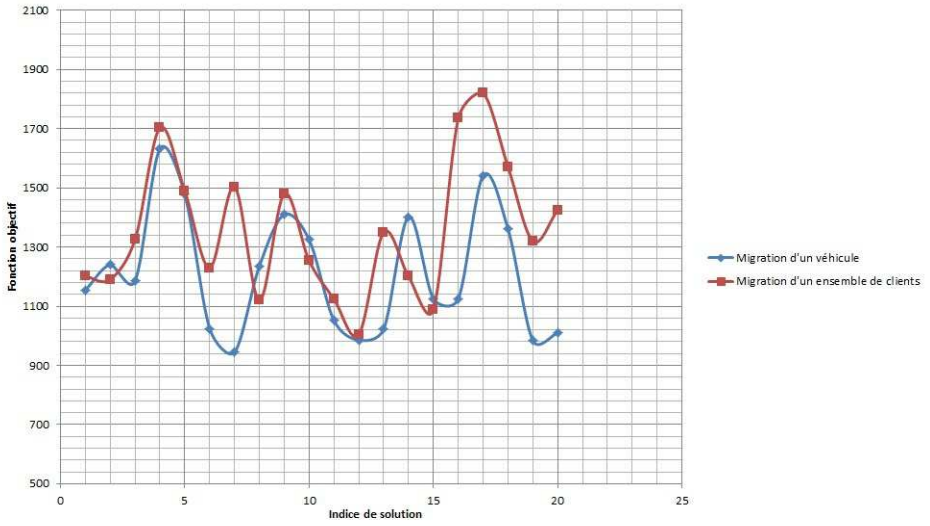

FIGURE 3.16 – Graphe de migration avec une population de 20 individus

L'algorithme qui résume l'adaptation de l'optimisation basée sur la biogéographie est présenté dans l'algorithme 25

3.5 Description du processus de coopération

Après avoir mis en oeuvre les trois métaheuristiques présentées dans les sections précédentes, la coopération peut être développée. Par la concurrence et la coopération, cette méthode va promouvoir la diversification de même que l'intensification de l'espace de recherche. Le schéma du processus est tel que présenté dans la section 3.2.

Dès qu'on commence la recherche, les métaheuristiques esclaves ont tous les trois un rang égal à zéro. Le choix de la métaheuristique de démarrage est aléatoire.

Lorsque le processus est lancé, la métaheuristique choisie aléatoirement mènera à une solution, le gain potentiel de qualité offert par cette nouvelle solution sera ajouté au rang de métaheuristique. Les rangs des métaheuristiques esclaves aideront à déterminer la prochaine métaheuristique à être exécuter.
La liste tabou comprendra les métaheuristiques qui ne pourraient pas améliorer la solution actuelle si l'amélioration survient pendant l'exécution d'algorithme, la liste taboue est vidée afin que tous les métaheuristiques esclaves seront libres à exécuter.
Le processus prend fin si les métaheuristiques n'ont pas pu amélioré la solution un nombre déterminé de fois consécutives.
Une fois la liste tabou est pleine, une métaheuristiques est aspirée à concurrencer les métaheuristiques en cours. Plusieurs critères d'aspiration peuvent être utilisés (la plus ancienne, la métaheuristique avec le meilleur rang ou au hasard). Nous avons opté pour un choix aléatoire entre ces options.

L'algorithme 26 résume le déroulement du processus de coopération.

3.6 Résultats de l'approche coopérative sur le VRPSPD

3.6.1 Jeux de données utilisé

L'ensemble de données utilisé est celui proposé par Salhi et Nagy [Salhi 1999]. Cet ensemble de données a été adapté à partir de sept problèmes initialement proposées par Christofides, Mingozzi, et Toth [Christofides 1979] pour le CVRP, impliquant 50 à 199 clients. La matrice de coût est obtenue en calculant la distance euclidienne entre les sommets. Salhi et Nagy [Salhi 1999] utilisent le schéma suivant pour modifier les instances de CVRP pour le VRPSPD : les coordonnées x_i et y_i d'un client i sont considérés comme dans le problème initial et un rapport r_i est calculé comme suit : $r_i = min(x_i/y_i, y_i/x_i)$. Soit d_{mi} la demande du client i. Pour obtenir les premières sept instances du VRPSPD, à savoir la série X, les quantités de livraison et collecte pour le client i sont calculées comme suit : $d_i = r_i \times dm_i$ et $p_i = (1 - r_i) \times dm_i$, respectivement. Les autres sept instances de VRPSPD, à savoir la série Y, sont générés en permutant les valeurs de d_i et p_i pour chaque client ([Gajpal 2009] ; [Subramanian 2010a] ; [Zachariadis 2010]). Les paramètres de chaque métaheuristique sont générés automatiquement au début du processus.

3.6.2 Étude comparative

Afin d'étudier les performances de l'approche proposée pour la VRPSPD, nous comparons les algorithmes suivants présentés dans la littérature :

- **ACS**(Ant Colony System) [Gajpal 2009].
- **PILS** (Parallel Iterative Local Search) [Subramanian 2010a].
- **BC** (Branch-and-Cut) [Subramanian 2010b].
- **BCP** (Branch-cut-and-price) [Subramanian 2013b].
- **h_PSO** (hybrid Particle Swarm Optimization)[Goksal 2013].
- **AMM**(Adaptive Memory Methodology)[Zachariadis 2010].

Les travaux récents de Subramanian et al. [Subramanian 2010a] sont ceux qui ont engendré les bornes supérieures les plus connues pour le VRPSPD. Tandis que les travaux de [Subramanian 2010b] et [Subramanian 2013b] ont donné les bornes inférieures les plus connues pour ce problème.

Les tableaux 3.1, 3.2, 3.3, 3.4, 3.5, 3.6 montrent les résultats de comparaison de l'approche proposée avec les algorithmes ACS, PILS, BC, BCP, h_PSO et AMM respectivement sur les instances de Salhi et Nagy [Salhi 1999]. La première colonne représente le nom de l'instance (chaque instance peut contenir de 50 jusqu'à 200 clients), la deuxième et la troisième colonne contiennent les résultats de l'approche ACS (AMM,PILS, h_PSO, BC ou BCP) et l'approche coopérative respectivement.La dernière colonne représente l'écart trouvé entre les solutions. Cet écart est calculé comme suit :

$$Ecart = \frac{Cout_{Approche} - Cout_{Cooperation}}{Cout_{Cooperation}} \times 100\% \tag{3.11}$$

A partir des tableaux, nous remarquons que l'approche coopérative a amélioré ACS de 0.19% (12 instances sur 14) et AMM de 0.04% (améliore 4 instances sur 14 et donne les mêmes résultats dans 7 instances sur 14). La moyenne d'écart entre les solutions de l'approche coopérative et PILS est de -0.062%. L'approche proposée n'a pu améliorer que 2 instances sur 14 et a donné les mêmes résultats que PILS dans 5 instances sur 14. Tandis qu'elle a pu améliorer les résultats de h_PSO de 0.03% (6 instances sur 14 et les mêmes résultats dans 5 instances sur 14). Nous remarquons aussi que l'écart entre les solutions de notre approche et BC et BCP est faible (-3.71% et -3.59%).

La figure 3.17 représente le coût des solutions des différentes approches en fonction des instances. Nous remarquons que les résultats de l'approche coopérative améliorent dans certains instances les méthodes ayant la meilleure borne supérieure. Et dans certains cas la différence est faible.

Instance	ACS	Coopération	Écart(%)
CMT1X	466.77	467	-0.04
CMT1Y	466.77	467	-0.04
CMT2X	684.21	684.21	0
CMT2Y	684.94	684.21	0.1
CMT3X	721.40	721.27	0.01
CMT12X	663.01	662.12	0.13
CMT3Y	721.40	719	0.33
CMT12Y	663.50	662.22	0.19
CMT11X	839.66	838.66	0.11
CMT11Y	840.19	837.08	0.37
CMT4X	854.12	852.46	0.19
CMT4Y	855.76	852.46	0.38
CMT5X	1034.87	1030.55	0.41
CMT5Y	1037.34	1030.55	0.65
Moyenne			**0.19**

TABLE 3.1 – Comparaison de la méthode de coopération avec ACS sur le problème de VRPSPSD

Instance	AMM	Coopération	Écart(%)
CMT1X	469.8	467	0.59
CMT1Y	469.8	467	0.59
CMT2X	684.21	684.21	0
CMT2Y	684.21	684.21	0
CMT3X	721.27	721.27	0
CMT12X	662.22	662.12	0.01
CMT3Y	721.27	719	0.31
CMT12Y	662.22	662.22	0
CMT11X	833.92	838.66	-0.56
CMT11Y	833.92	837.08	-0.37
CMT4X	852.46	852.46	0
CMT4Y	852.46	852.46	0
CMT5X	1030.55	1030.55	0
CMT5Y	1030.55	1030.55	0
Moyenne			**0.04**

TABLE 3.2 – Comparaison de la méthode de coopération avec AMM sur le problème de VRPSPSD

Instance	PILS	Coopération	Écart(%)
CMT1X	466.77	467	-0.04
CMT1Y	466.77	467	-0.04
CMT2X	684.21	684.21	0
CMT2Y	684.21	684.21	0
CMT3X	721.27	721.27	0
CMT12X	662.22	662.12	0.01
CMT3Y	721.27	719	0.31
CMT12Y	662.22	662.22	0
CMT11X	833.92	838.66	-0.56
CMT11Y	833.92	837.08	-0.37
CMT4X	852.46	852.46	0
CMT4Y	852.46	852.46	0
CMT5X	1029.55	1030.55	-0.09
CMT5Y	1029.55	1030.55	-0.09
Moyenne			**-0.062**

TABLE 3.3 – Comparaison de la méthode de coopération avec PILS sur le problème de VRPSPSD

Instance	h_PSO	Coopération	Écart(%)
CMT1X	466.77	467	-0.04
CMT1Y	466.77	467	-0.04
CMT2X	684.21	684.21	0
CMT2Y	684.21	684.21	0
CMT3X	721.27	721.27	0
CMT12X	662.94	662.12	0.12
CMT3Y	721.27	719	0.31
CMT12Y	663.50	662.22	0.19
CMT11X	833.92	838.66	-0.56
CMT11Y	833.92	837.08	-0.37
CMT4X	852.83	852.46	0.04
CMT4Y	852.46	852.46	0
CMT5X	1033.50	1030.55	0.28
CMT5Y	1036	1030.55	0.52
Moyenne			**0.03**

TABLE 3.4 – Comparaison de la méthode de coopération avec h_PSO sur le problème de VRPSPSD

Instance	BC	Coopération	Écart(%)
CMT1X	460.73	467	-1.34
CMT1Y	460.69	467	-1.35
CMT2X	658.38	684.21	-3.77
CMT2Y	658.12	684.21	-3.81
CMT3X	711.77	721.27	-1.31
CMT12X	635.52	662.12	-4.01
CMT3Y	711.89	719	-0.98
CMT12Y	635.45	662.22	-4.01
CMT11X	793.11	838.66	-5.43
CMT11Y	793.54	837.08	-5.20
CMT4X	826.74	852.46	-3.01
CMT4Y	826.34	852.46	-3.06
CMT5X	971.07	1030.55	-5.77
CMT5Y	937.66	1030.55	-9.01
Moyenne			**-3.71**

TABLE 3.5 – Comparaison de la méthode de coopération avec BC sur le problème de VRPSPSD

Instance	BCP	Coopération	Écart(%)
CMT1X	461.90	467	-1.09
CMT1Y	460.86	467	-1.31
CMT2X	661.11	684.21	-3.37
CMT2Y	659.76	684.21	-3.57
CMT3X	711.7	721.27	-1.32
CMT12X	638.72	662.12	-3.53
CMT3Y	711.49	719	-1.04
CMT12Y	636.29	662.22	-3.39
CMT11X	766.22	838.66	-8.63
CMT11Y	765.47	837.08	-8.55
CMT4X	829.58	852.46	-2.68
CMT4Y	827.50	852.46	-2.92
CMT5X	988.28	1030.55	-4.10
CMT5Y	980.63	1030.55	-4.84
Moyenne			**-3.59**

TABLE 3.6 – Comparaison de la méthode de coopération avec BCP sur le problème de VRPSPSD

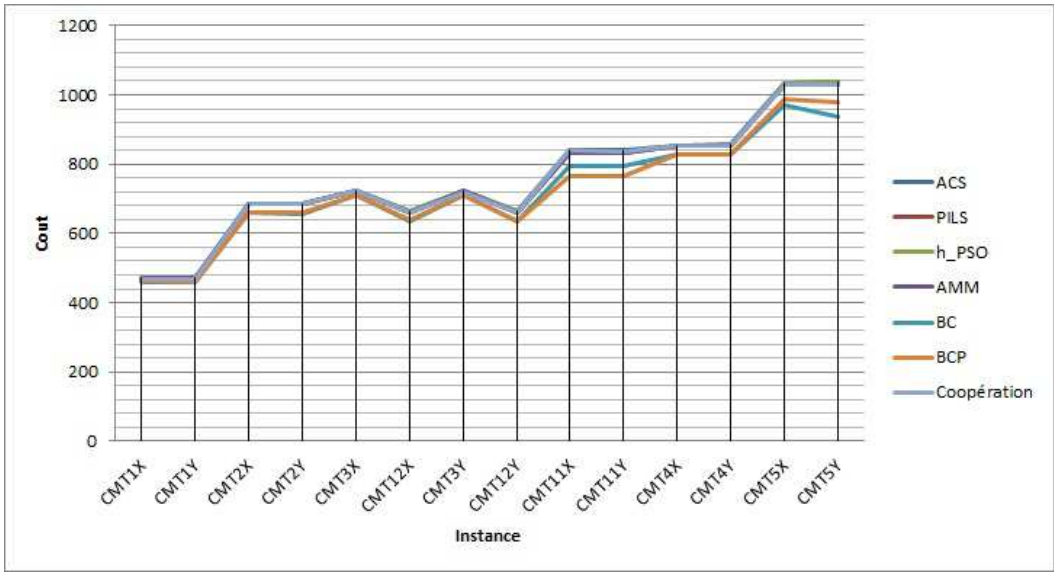

FIGURE 3.17 – Coûts des solutions en fonction des instances

3.7 Conclusion

Ce chapitre vise à résoudre grâce à une approche coopérative le problème de tournées de véhicules avec collecte et livraison simultanées VRPSPD. L'approche proposée est une méthode de coopération entre les trois métaheuristiques (le recuit simulé, l'algorithme génétique et l'optimisation basée sur la biogéographie) supervisés par une métaheuristiques maitre (recherche tabou). L'expérimentation a montré que dans la plupart des cas, une amélioration dans la qualité des solutions est fournie par la coopération. En effet, celle-ci bénéficie des avantages de chacune de ces méthodes. Chacune des métaheuristques sortante de la compétition, fournit sa meilleure solution à la métaheuristique qui la remplace. Ceci rend le processus de recherche des solutions plus efficace.

Algorithme 21 Algorithme de croisement

Sélectionner deux solutions S_i,S_j à partir de la population
Déterminer aléatoirement la position de coupure de S_i et de S_j
Croiser(S_i,S_j)
Corriger les deux nouvelles solutions enfants S_{ij} et S_{ji}
si S_{ij} et $S_{ji} \notin$ *population* **alors**
  Ajouter S_{ij} et S_{ji} à la population

Algorithme 22 Algorithme de Mutation dans l'AG

ENTREE : solution aléatoire S_i de la population
Appliquer la mutation de permutation sur un véhicule
Appliquer la mutation de permutation entre deux véhicules

Algorithme 23 Adaptation Algorithme Génétique

Générer aléatoirement une population de solutions initiales $S_0, S_1 ... S_n$
Évaluer la fitness de chaque solution
$i = 0$
tant que $i < taille_{Population}$ et condition d'arrêt non satisfaite **faire**
  Croisement :
  tant que $rand_{croisement}(0;1) < taux_{croisement}$ **faire**
    Sélectionner aléatoirement les parents S_j et S_k dans la population selon la stratégie choisie
    Croiser les parents pour obtenir deux nouvelles solutions enfants S_j^* et S_k^*
    Ajouter les enfants S_j^* et S_k^* à la population
  Mutation :
  tant que $random_{mutation}(0-1) < taux_{mutation}$ **faire**
    Sélectionner aléatoirement une solution S_j selon la stratégie choisie
    Muter la solution S_j en $S_j^{'}$
    Ajouter la solution $S_j^{'}$ à la population
  Réduire la taille de la population à $taille_{Population}$ en ne gardant que les meilleurs individus
  $i = i + 1$
Retourner le meilleur individu trouvé

Algorithme 24 Algorithme de migration

ENTRÉE : deux solutions S_i,S_j à partir de la population
Chercher V_{min} dans Si
Migrer V_{min} vers Sj
Corriger Sj

Algorithme 25 Algorithme d'adaptation de l'optimisation basée sur la biogéographie

Générer aléatoirement une population de solutions initiales $S_0, S_1 ... S_n$
Évaluer la fitness de chaque solution
$i = 0$
tant que $i < nombre_{gneration}$ **faire**
  Calculer le nombre d'espèce, le taux d'immigration λ et d'émigration μ pour chaque solution
  Migration :
  pour chaque S_i **faire**
    si $rand_{immigration}(0;1) < \lambda$ **alors**
      Sélectionner S_i
      pour chaque S_j **faire**
        si $rand_{emmigration}(0;1) < \mu$ **alors**
          Sélectionner S_j
          Migration(S_j,S_i)
  Mutation :
  pour chaque solution S_i **faire**
    Muter(S_i)
Retourner le meilleur individu trouvé ;

Algorithme 26 Algorithme de coopération des métaheuristiques

$similaire := 100$;
Choisir aléatoirement $meta$ parmi $[AG, RS, BBO]$
tant que $similaire > 0$ **faire**
  Appliquer $meta$
  Calculer $Amelioration_{meta}$
  $rang_{meta} = Amelioration_{meta}$
  tant que $Amelioration_{meta} < 0$ **faire**
    $similaire = similaire - 1$
    choisir $meta$ qui a le plus grand rang
    Appliquer $meta$
    Calculer $Amelioration_{meta}$
    $rang_{meta} = rang_{meta} + Amelioration_{meta}$
  $similaire = 100$
  Ajouter $meta$ à liste taboue
  si liste pleine **alors**
    Choisir aléatoirement $meta$ parmi $[AG, RS, BBO]$ et la libérer
  sinon
    Choisir $meta$ qui a le plus grand rang

Chapitre 4

Conclusion et Perspectives

Ce chapitre clôt la thèse en rappelant les objectifs et les points abordés. Les travaux présentés dans cette thèse traitent le fameux problème de tournées de véhicules qui malgré son ancienneté, reste l'un des problèmes les plus traités dans le domaine de l'optimisation combinatoire.
Les problèmes de tournées de véhicules (Vehicle Routing Problems) sont une grande famille de problèmes d'optimisation combinatoire avec des applications dans de nombreux domaines différents, allant de la distribution de biens à la livraison des services.

Dans cette thèse, nous avons étudié deux variantes du problème de tournées de véhicules avec collecte et livraison, à savoir, le problème de tournées de véhicules avec flotte hétérogène, collecte et livraison et fenêtres de temps (HVRPMBTW) et le problème de tournées de véhicules avec collecte et livraison simultanées (VRPSPD). Le HVRPMBTW consiste à servir deux types de clients (clients de livraison et clients de collecte) sans contrainte de précédence en utilisant une flotte hétérogène et limitée de véhicules de coûts différents. Chaque client doit être servi dans un intervalle de temps bien précis (fenêtre de temps). Le problème VRPSPD consiste à servir un ensemble de clients avec une flotte homogène et illimitée de véhicules. Chaque client étant un point de collecte et de livraison.

L'objectif était de prospecter les voies d'adaptation et de coopération dans les métaheuristiques pour résoudre les deux problèmes cités. Dans la première partie de la thèse, pour résoudre la variante HVRPMBTW, nous avons proposé quatre approches de résolution en utilisant l'aspect d'adaptation. Cet aspect a été utilisé pour le réglage automatique des paramètres des métaheuristiques utilisées. Ce réglage est très important pour la qualité des résultats obtenus. Toutes les approches proposées passent par une phase d'adaptation des paramètres aux instances de problème traité. Nous avons testé nos approches sur différents ensembles de problème dérivés de l'ensemble de données de Solomon [Solomon 1987] $R1, R2, C1, C2, RC1$ et $RC2$. Ces données contiennent 55 instances de VRPTW, chacune contient 100 clients. Les résultats sont comparés avec les méthodes ACO (algorithme de colonies de fourmis)et PSO (optimisation par essaims des particules) de Belmecheri et al. [Belmecheri 2013].
La première approche consiste en l'amélioration de l'algorithme BBO (Biogeography Based Optimisation). Ce dernier est l'un des algorithmes récents basé

sur les mathématiques de biogéographie. La nouvelle approche est nommée EBBO (an Enhanced Biogeography Based Optimization algorithm). L'amélioration est effectuée par l'application de l'algorithme recuit simulé à chaque solution dans chaque itération. L'approche proposée EBBO a amélioré les résultats d'ACO dans 31 instances sur 55, et les résultats de PSO dans 28 instances sur 55.
La deuxième approche utilise une nouvelle métaheuristique inspirée du comportement de coucous nommée CS (Cuckoo Search). Cette dernière étant conçu pour des problèmes continus, nous avons dû adapter une version binaire discrète, nommée DBCS au problème. L'approche proposée DBCS a amélioré ACO dans 32 instances sur 55 et les résultats de PSO dans 21 instances sur 55.
La troisième approche proposée nommée EDHS est l'algorithme amélioré discret de la recherche d'harmonie qui est un algorithme amélioré et discrétisé de l'algorithme standard HS (Harmony Search). EDHS a amélioré ACO dans 23 instances sur 55 et les résultats de PSO dans 21 instances sur 55.
Dans la dernière approche, nous proposons d'utiliser un algorithme inspiré de l'informatique quantique avec une taille de population variable QIHSVPS. Nous avons choisi de réduire la taille de la population lorsque l'amélioration de la qualité de la solution est insignifiante. Les principes de l'informatique quantique sont utilisés pour accélérer le processus d'évolution alors que la taille de la population variable est utilisée pour diminuer le nombre d'évaluations de la solution, lorsque l'amélioration est négligeable. L'approche QIHSVPS (Algorithme quantique de recherche d'harmonie avec taille de population variable) a amélioré les résultats d'ACO dans 40 instances sur 55 et les résultats de PSO dans 36 instances sur 55. L'analyse statistique des six approches a montré que l'hybridation avec les principes quantiques permet d'améliorer la qualité de la solution.

Dans la seconde partie de la thèse, nous avons proposé une approche coopérative pour résoudre le VRPSPD. L'approche proposée est une méthode de coopération entre trois métaheuristiques (Recuit simulé, Algorithmes génétiques et Optimisation basée sur la biogéographie) supervisées par une métaheuristique maitre (Recherche taboue). L'approche coopérative a été testée sur l'ensemble de données proposé par Salhi et Nagy [Salhi 1999]. Pour évaluer la performance de l'approche coopérative, nous avons comparé avec des algorithmes récents présentés dans la littérature (Ant Colony System ACS [Gajpal 2009], Parallel Iterative Local Search PILS [Subramanian 2010a], Branch-and-Cut BC [Subramanian 2010b], Branch-cut-and-price BCP [Subramanian 2013b], hybrid Particle Swarm Optimization h_PSO [Goksal 2013], Adaptive Memory Methodology AMM[Zachariadis 2010]). L'expérimentation a révélé, dans tous les cas, une amélioration de la qualité de la solution apportée par la coopéra-

tion. En effet, L'approche coopérative a amélioré ACS dans 12 instances sur 14, et amélioré AMM dans 7 instances sur 14 (4 résultats sont égaux), tandis qu'elle a amélioré PILS dans 2 instances sur 14 (5 résultats sont égaux) et h_PSO dans 6 instances sur 14 (4 résultats sont égaux). Il est à noter que l'écart entre les solutions de notre approche et BC et BCP est faible (-3.71% et -3.59%).

Comme perspectives, nous aimerions pouvoir tester nos approches sur un cas réel de distribution et collecte de biens. Nous aimerions aussi traiter des variantes modélisées comme des problèmes multi-objectifs, par exemple les problèmes ayant pour objectif de minimiser la distance et maximiser le profit tiré des clients ou de minimiser le nombre de véhicules utilisé. Nous aimerions aussi prospecter la voie de la parallélisation.

Bibliographie

[Baldacci 2008] Roberto Baldacci, Nicos Christofides et Aristide Mingozzi. *An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation with Additional Cuts*. Math. Program., vol. 115, no. 2, pages 351–385, 2008. (Cité en page 17.)

[Baldacci 2012] Roberto Baldacci, Aristide Mingozzi et Roberto Roberti. *Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints*. European Journal of Operational Research, vol. 218, no. 1, pages 1–6, 2012. (Cité en page 17.)

[Belmecheri 2013] Farah Belmecheri, Christian Prins, Farouk Yalaoui et Lionel Amodeo. *Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows*. J. Intelligent Manufacturing, vol. 24, no. 4, pages 775–789, 2013. (Cité en pages 23, 48, 49, 50, 55 et 89.)

[Bent 2006] Russell Bent et Pascal Van Hentenryck. *A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows*. Computers & Operations Research, vol. 33, no. 4, pages 875–893, 2006. (Cité en page 20.)

[Berghida] Meryem Berghida et Abdelmadjid Boukra. *Resolving a Vehicle Routing Problem with Heterogeneous Fleet, Mixed Backhauls and Time Windows using Cuckoo Behavior Approach.* International Journal of Operational Research. (Cité en page 39.)

[Berghida 2014a] Meryem Berghida et Abdelmadjid Boukra. *EBBO : an enhanced biogeography-based optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows*. The International Journal of Advanced Manufacturing Technology, pages 1–15, 2014. (Cité en page 32.)

[Berghida 2014b] Meryem Berghida et Abdelmadjid Boukra. *EDHSA : An enhanced discrete harmony search algorithm for a Vehicle Routing Problem with Heterogeneous Fleet, Mixed Backhauls, and Time Windows*. In ISIA, 2014. (Cité en page 42.)

[Berghida 2014c] Meryem Berghida, Abdelmadjid Boukra, Hani Amar Guenoune et Mohammed Merouane El Ksouri. *Cooperative Approach for Vehicle Routing Problem with Simultaneous Pickup and Delivery*. In 3th International Symposium on Modeling and Implementation of Complex Systems, 2014. (Cité en page 59.)

[Berhan 2014] Eshetie Berhan, Pavel Krömer, Daniel Kitaw, Ajith Abraham et Václav Snášel. *Solving Stochastic Vehicle Routing Problem with Real Simultaneous Pickup and Delivery Using Differential Evolution*. In Ajith Abraham, Pavel Krömer et Václav Snášel, editeurs, Innovations in Bio-inspired Computing and Applications SE - 18, volume 237 of *Advances in Intelligent Systems and Computing*, pages 187–200. Springer International Publishing, 2014. (Cité en page 59.)

[Bettinelli 2011] Andrea Bettinelli, Alberto Ceselli et Righini Giovanni. *A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows*. Transportation Research Part C : Emerging Technologies, vol. 19, no. 5, pages 723–740, 2011. (Cité en page 18.)

[Brest 2015] Janez Brest, Aleš Zamuda et Borko Bošković. *Adaptation in the Differential Evolution*. In Iztok Fister et Iztok Fister Jr., editeurs, Adaptation and Hybridization in Computational Intelligence SE - 2, volume 18 of *Adaptation, Learning, and Optimization*, pages 53–68. Springer International Publishing, 2015. (Cité en page 26.)

[Brown 2007] Brown, Clifford, Liebovitch, Larry, Glendon et Rachel. *Levy Flights in Dobe Ju/hoansi Foraging Patterns*. Human Ecology, no. 1, pages 129–138, Février 2007. (Cité en page 37.)

[Brownlee 2011] J Brownlee. Clever Algorithms : Nature-Inspired Programming Recipes. Lulu Enterprises Incorporated, 2011. (Cité en page 70.)

[Cao 2014] Erbao Cao, Mingyong Lai et Hongming Yang. *Open vehicle routing problem with demand uncertainty and its robust strategies*. Expert Systems with Applications, vol. 41, no. 7, pages 3569–3575, 2014. (Cité en page 20.)

[Chen 2007] Ping Chen, Houkuan Huang et Xingye Dong. *An Ant Colony System Based Heuristic Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pickup*. In Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on, pages 136–141, 2007. (Cité en page 58.)

[Chen 2010] Jianxin Chen, Qing Liu, Junqin Huang et Yun Hou. *An Adaptive Genetic Algorithm Based on Multi-population Parallel Evolutionary and Variable Population Size*. 2010 Second WRI Global Congress on Intelligent Systems, vol. 1, pages 258–262, 2010. (Cité en page 44.)

[Chen 2012a] R.-M. Chen, F.-R. Hsieh et D.-S. Wu. *Heuristics based ant colony optimization for vehicle routing problem*. In Proceedings of the 2012 7th IEEE Conference on Industrial Electronics and Applications, ICIEA 2012, pages 1039–1043, 2012. (Cité en page 19.)

[Chen 2012b] Yi Chen et Zhi-Jun Song. *Spatial analysis for functional region of suburban-rural area using micro genetic algorithm with variable population size.* Expert Syst. Appl., vol. 39, no. 7, pages 6469–6475, 2012. (Cité en page 44.)

[Christofides 1976] Nicos Christofides. *The vehicle routing problem.* RAIRO - Operations Research - Recherche Opérationnelle, vol. 10, no. V1, pages 55–70, 1976. (Cité en page 7.)

[Christofides 1979] N. Christofides, A. Mingozzi et P. Toth. *The vehicle routing problem.* In N. Christofides, A. Mingozzi, P. Toth et L. Sandi, editeurs, Combinatorial optimization, pages 315–338. Wiley, Chichester, UK, 1979. (Cité en page 81.)

[coo 2014] *Cooperative Coevolution.* In Support Vector Machines and Evolutionary Algorithms for Classification SE - 5, volume 69 of *Intelligent Systems Reference Library*, pages 57–73. Springer International Publishing, 2014. (Cité en page 61.)

[Cordeau 2001] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon et F. Soumis. *The Vehicle Routing Problem.* chapitre VRP with Time Windows, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. (Cité en page 9.)

[Dantzig 1959] G B Dantzig et J H Ramser. *The Truck Dispatching Problem.* Management Science, vol. 6, pages 80–91, Octobre 1959. (Cité en page 5.)

[Darwin 1995] Charles Darwin. The origin of species. New York :P.F. Collier„ 1995. http ://www.biodiversitylibrary.org/bibliography/24252. (Cité en page 28.)

[Desaulniers 2001] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon et F. Soumis. *The Vehicle Routing Problem.* chapitre VRP with Pickup and Delivery, pages 225–242. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. (Cité en page 13.)

[Dethloff 2001] Jan Dethloff. *Vehicle routing and reverse logistics : The vehicle routing problem with simultaneous delivery and pick-up.* OR-Spektrum, vol. 23, no. 1, pages 79–96, 2001. (Cité en page 58.)

[Fisher 1981] Marshall L. Fisher et Ramchandran Jaikumar. *A generalized assignment heuristic for vehicle routing.* Networks, vol. 11, no. 2, pages 109–124, 1981. (Cité en page 10.)

[Fister Jr. 2015] Iztok Fister Jr. et Iztok Fister. *On the Mutation Operators in Evolution Strategies.* In Iztok Fister et Iztok Fister Jr., editeurs, Adaptation and Hybridization in Computational Intelligence SE - 3,

volume 18 of *Adaptation, Learning, and Optimization*, pages 69–89. Springer International Publishing, 2015. (Cité en page 27.)

[Fleurent 1994] Charles Fleurent et Jacques A Ferland. *Algorithmes Génétiques Hybrides Pour L'optimisation Combinatoire*, 1994. (Cité en page 61.)

[Franceschelli 2013] Mauro Franceschelli, Daniele Rosa, Carla Seatzu et Francesco Bullo. *Gossip algorithms for heterogeneous multi-vehicle routing problems*. Nonlinear Analysis : Hybrid Systems, vol. 10, no. 0, pages 156–174, 2013. (Cité en page 19.)

[Friedman 1937] Milton Friedman. *The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance*. Journal of the American Statistical Association, vol. 32, no. 200, pages 675–701, 1937. (Cité en page 55.)

[Gajpal 2009] Yuvraj Gajpal et Prakash Abad. *An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup*. Computers & Operations Research, vol. 36, no. 12, pages 3215–3223, Décembre 2009. (Cité en pages 81, 82 et 90.)

[Geem 2001] Zong Woo Geem, Joong Hoon Kim et G V Loganathan. *A new heuristic optimization algorithm : harmony search*. Simulation, vol. 76, no. 2, pages 60–68, 2001. (Cité en page 41.)

[Geem 2005] Zong Woo Geem, Kang Seok Lee et Yong Jin Park. *Application of Harmony Search to Vehicle Routing*. American Journal of Applied Sciences, vol. 2, no. 12, pages 1552–1557, 2005. (Cité en page 18.)

[Goksal 2010] F P Goksal, F Altiparmak et I Karaoglan. *A hybrid particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery*. In Computers and Industrial Engineering (CIE), 2010 40th International Conference on, pages 1–6, 2010. (Cité en page 58.)

[Goksal 2013] Fatma Pinar Goksal, Ismail Karaoglan et Fulya Altiparmak. *A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery*. Computers & Industrial Engineering, vol. 65, no. 1, pages 39–53, Mai 2013. (Cité en pages 82 et 90.)

[Goldberg 1989] David E. Goldberg. Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st édition, 1989. (Cité en page 72.)

[Grellier 2008] Emilie Grellier. *Optimisation de tournées de véhicules dans le cadre de la logistique inverse : modélisation et résolution par des*

méthodes hybrides. These, Université de Nantes, Janvier 2008. (Cité en page 10.)

[Gribkovskaia 2007] Irina Gribkovskaia, Ø yvind Halskau Sr., Gilbert Laporte et Martin Vlček. *General solutions to the single vehicle routing problem with pickups and deliveries*. European Journal of Operational Research, vol. 180, no. 2, pages 568–584, 2007. (Cité en page 18.)

[Gulic 2013] M Gulic et D Jakobovic. *Evolution of vehicle routing problem heuristics with genetic programming*. In Information Communication Technology Electronics Microelectronics (MIPRO), 2013 36th International Convention on, pages 988–992, 2013. (Cité en page 20.)

[Haghani 2005] Ali Haghani et Soojung Jung. *A Dynamic Vehicle Routing Problem with Time-dependent Travel Times*. Comput. Oper. Res., vol. 32, no. 11, pages 2959–2986, Novembre 2005. (Cité en page 16.)

[Holland 1962] John H. Holland. *Outline for a Logical Theory of Adaptive Systems*. J. ACM, vol. 9, no. 3, pages 297–314, Juillet 1962. (Cité en page 72.)

[Holland 1975] J H Holland. Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975. (Cité en page 25.)

[Hong 2012] Lianxi Hong. *An improved {LNS} algorithm for real-time vehicle routing problem with time windows*. Computers & Operations Research, vol. 39, no. 2, pages 151–163, 2012. (Cité en page 19.)

[Hu 2009] Feng-jun Hu et Bin Wu. *Quantum evolutionary algorithm for vehicle routing problem with simultaneous delivery and pickup*. In Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, pages 5097–5101, 2009. (Cité en page 18.)

[Julstrom 2005] Bryant A. Julstrom. *Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem.* In Hans-Georg Beyer et Una-May O'Reilly, editeurs, GECCO, pages 607–614. ACM, 2005. (Cité en page 38.)

[Khouadjia 2010] Mostepha Redouane Khouadjia, Laetitia Jourdan et El-Ghazali Talbi. *Adaptive particle swarm for solving the Dynamic Vehicle Routing Problem.* In AICCSA, pages 1–8. IEEE, 2010. (Cité en page 16.)

[Kirkpatrick 1983a] S Kirkpatrick, C D Gelatt et M P Vecchi. *Optimization by simulated annealing*. Science, vol. 220, pages 671–680, 1983. (Cité en page 35.)

[Kirkpatrick 1983b] S Kirkpatrick, C D Gelatt et M P Vecchi. *Optimization by simulated annealing.* Science, vol. 220, pages 671–680, 1983. (Cité en page 69.)

[Kong 2006] Min Kong et Peng Tian. *Apply the Particle Swarm Optimization to the Multidimensional Knapsack Problem.* In Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh et Jacek M. Zurada, editeurs, Artificial Intelligence and Soft Computing - ICAISC 2006, 8th International Conference, Zakopane, Poland, June 25-29, 2006, Proceedings, volume 4029 of *Lecture Notes in Computer Science*, pages 1140–1149. Springer, 2006. (Cité en page 38.)

[Laporte 1992] G Laporte. *The vehicle routing problem : An overview of exact and approximate algorithms.* European Journal of Operational Research, vol. 59, no. 3, pages 345–358, 1992. (Cité en pages 2 et 17.)

[Liao 2007] Ching-Jong Liao, Chao-Tang Tseng et Pin Luarn. *A discrete version of particle swarm optimization for flowshop scheduling problems.* Comput. Oper. Res., vol. 34, no. 10, pages 3099–3111, Octobre 2007. (Cité en page 38.)

[Liao 2012] Xin-Lan Liao et Chuan-Kang Ting. *Evolutionary algorithms using adaptive mutation for the selective pickup and delivery problem.* In Evolutionary Computation (CEC), 2012 IEEE Congress on, pages 1–8, 2012. (Cité en page 19.)

[Lin 2014] Canhong Lin, K L Choy, G T S Ho, S H Chung et H Y Lam. *Survey of Green Vehicle Routing Problem : Past and future trends.* Expert Systems with Applications, vol. 41, no. 4, Part 1, pages 1118–1138, 2014. (Cité en page 17.)

[Liu 2010] Chang-shi Liu et Qi-Jin Tang. *A hybrid heuristics for vehicle routing problem with simultaneous pickup and delivery service.* In Logistics Systems and Intelligent Management, 2010 International Conference on, volume 3, pages 1422–1426, Janvier 2010. (Cité en page 58.)

[Liu 2013a] Ran Liu, Xiaolan Xie, Vincent Augusto et Carlos Rodríguez-Verjan. *Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care.* European Journal of Operational Research, vol. 230, no. 3, pages 475–486, 2013. (Cité en page 19.)

[Liu 2013b] Shuguang Liu. *A hybrid population heuristic for the heterogeneous vehicle routing problems* . Transportation Research Part E : Logistics and Transportation Review, vol. 54, no. 0, pages 67–78, 2013. (Cité en page 21.)

[MacArthur 1967] R.H. MacArthur et E.O. WILSON. The theory of island biogeography. Monographs in population biology. Princeton University Press, 1967. (Cité en page 28.)

[Macedo 2011] Rita Macedo, Cláudio Alves, J M Valério de Carvalho, François Clautiaux et Saïd Hanafi. *Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model*. European Journal of Operational Research, vol. 214, no. 3, pages 536–545, 2011. (Cité en page 18.)

[MacQueen 1967] J MacQueen. *Some methods for classification and analysis of multivariate observations*, 1967. (Cité en page 64.)

[Marinakis 2012] Yannis Marinakis. *Multiple Phase Neighborhood Search-GRASP for the Capacitated Vehicle Routing Problem*. Expert Systems with Applications, vol. 39, no. 8, pages 6807–6815, 2012. (Cité en page 19.)

[Marinakis 2013] Yannis Marinakis et Magdalene Marinaki. *A Bumble Bees Mating Optimization algorithm for the Open Vehicle Routing Problem*. Swarm and Evolutionary Computation, no. 0, pages –, 2013. (Cité en page 14.)

[Meng 2008] Lijun Meng et Xiaochai Guo. *A new hybrid metaheuristics for the vehicle routing problem with simultaneous pick-up and delivery*. In Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008. IEEE International Conference on, volume 1, pages 1198–1202, Octobre 2008. (Cité en page 58.)

[Metropolis 1953] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller et Edward Teller. *Equation of State Calculations by Fast Computing Machines*. The Journal of Chemical Physics, vol. 21, no. 6, pages 1087–1092, 1953. (Cité en page 69.)

[Min 1989] H Min. *The multiple vehicle routing problem with simultaneous delivery and pick-up points*. Transportation Research Part A : General, vol. 23, no. 5, pages 377–386, 1989. (Cité en page 58.)

[Minetti 2005] Gabriela F Minetti et Hugo Alfonso. *Variable Size Population in Parallel Evolutionary Algorithms*. In ISDA, pages 350–355. IEEE Computer Society, 2005. (Cité en page 44.)

[MITRE 2005] L.B.P.S. MITRE, A.S.F.P.C.S.U.N. Mexico, O.G.I.S.S.E.O.H.S.U.M.M.A.P.C.S. Engineering et A A of the Computing Lab at the Center for the Study of Complex Systems University of Michigan Rick Riolo Research Scientist. Perspectives on Adaptation in Natural and Artificial Systems. Santa Fe Institute

Studies on the Sciences of Complexity. Oxford University Press, USA, 2005. (Cité en page 26.)

[Montané 2006] Fermín Alfredo Tang Montané et Roberto Diéguez Galvão. *A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service*. Computers & Operations Research, vol. 33, no. 3, pages 595–619, 2006. (Cité en pages 59 et 63.)

[Nagy 2005] Gábor Nagy et Saad Salhi. *Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries*. European Journal of Operational Research, vol. 162, no. 1, pages 126–141, 2005. (Cité en pages 18 et 58.)

[Naji-Azimi 2013] Zahra Naji-Azimi et Majid Salari. *A complementary tool to enhance the effectiveness of existing methods for heterogeneous fixed fleet vehicle routing problem*. Applied Mathematical Modelling, vol. 37, no. 6, pages 4316–4324, 2013. (Cité en page 19.)

[Ombuki 2006] Beatrice Ombuki, Brian J. Ross et Franklin Hanshar. *Multi-objective Genetic Algorithms for Vehicle Routing Problem with Time Windows*. APPLIED INTELLIGENCE, vol. 24, page 2006, 2006. (Cité en page 65.)

[Panneerselvam 2012] Suresh Nanda Kumar ; Ramasamy Panneerselvam. *A Survey on the Vehicle Routing Problem and Its Variants*. Intelligent Information Management, vol. 4, no. 3, pages 66–74, 2012. (Cité en page 17.)

[Parragh 2008] Sophie Parragh, Karl Doerner et Richard Hartl. *A survey on pickup and delivery problems*. Journal für Betriebswirtschaft, vol. 58, no. 2, pages 81–117, 2008. (Cité en page 17.)

[Pavlyukevich 2007] Ilya Pavlyukevich. *Levy flights, non-local search and simulated annealing*. J. Comput. Phys., vol. 226, no. 2, pages 1830–1844, Octobre 2007. (Cité en page 37.)

[Pillac 2013] Victor Pillac, Michel Gendreau, Christelle Guéret et Andrés L Medaglia. *A review of dynamic vehicle routing problems.* European Journal of Operational Research, vol. 225, no. 1, pages 1–11, 2013. (Cité en page 17.)

[Potter 1994] Mitchell A Potter et Kenneth A De Jong. *A Cooperative Coevolutionary Approach to Function Optimization*. In Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature : Parallel Problem Solving from Nature, PPSN III, pages 249–257, London, UK, UK, 1994. Springer-Verlag. (Cité en page 61.)

[Preux 1999] P Preux et E.-G. Talbi. *Towards hybrid evolutionary algorithms*. International Transactions in Operational Research, vol. 6, no. 6, pages 557–570, 1999. (Cité en page 61.)

[Psaraftis 1980] Harilaos N Psaraftis. *A dynamic-programming solution to the single vehicle many to many immediate request dial-a-ride problem.* Transportation Science, vol. 14, no. 2, pages 130–154, 1980. (Cité en page 16.)

[Psaraftis 1995] Harilaos N Psaraftis. *Dynamic vehicle routing : Status and prospects.* Annals of Operations Research, vol. 61, no. 1, pages 143–164, 1995. (Cité en page 16.)

[Ralphs 2003] Ted K. Ralphs, L. Kopman, William R. Pulleyblank et Leslie E. Trotter Jr. *On the capacitated vehicle routing problem.* Math. Program., vol. 94, no. 2-3, pages 343–359, 2003. (Cité en page 9.)

[Rego 1994] C Rego, C Roucairol et Institut National de recherche en informatique et en automatique. Le probl{è}me de tourn{é}es de v{é}hicules : {é}tude et r{é}solution approch{é}e. Rapports de recherche. Institut national de recherche en informatique et en automatique, 1994. (Cité en page 7.)

[Reynolds 2007] A.M. Reynolds et M.A. Frye. *Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search.* PLoS One, vol. 2, no. 4, page e354, 2007. (Cité en page 37.)

[Rieck 2013] Julia Rieck et Jürgen Zimmermann. *Exact Solutions to the Symmetric and Asymmetric Vehicle Routing Problem with Simultaneous Delivery and Pick-Up*. BuR - Business Research, vol. 6, no. 1, pages 77–92, 2013. (Cité en page 58.)

[Rochat 1995] Y. Rochat et É.D. Taillard. *Probabilistic diversification and intensification in local search for vehicle routing.* Journal of Heuristics, vol. 1, no. 1, pages 147–167, 1995. (Cité en page 16.)

[Ropke 2006] Stefan Ropke et David Pisinger. *A unified heuristic for a large class of vehicle routing problems with backhauls*. European Journal of Operational Research, vol. 2004, pages 750–775, 2006. (Cité en page 18.)

[Salhi 1999] S Salhi et G Nagy. *A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling*. Journal of the Operational Research Society, vol. 50, no. 10, pages 1034–1042, 1999. (Cité en pages 81, 82 et 90.)

[Sariklis 2000] D Sariklis et S Powell. *A heuristic method for the open vehicle routing problem.* Journal of the Operational Research Society, vol. 51, no. 5, pages 564–573, 2000. (Cité en page 14.)

[Simon 2008] Dan Simon. *Biogeography-Based Optimization*. IEEE Trans. Evolutionary Computation, vol. 12, no. 6, pages 702–713, 2008. (Cité en page 28.)

[Singh 2007] Alok Singh et Anurag Singh Baghel. *A New Grouping Genetic Algorithm for the Quadratic Multiple Knapsack Problem.* In Carlos Cotta et Jano I. van Hemert, editeurs, EvoCOP, volume 4446 of *Lecture Notes in Computer Science*, pages 210–218. Springer, 2007. (Cité en page 38.)

[Solomon 1987] Marius M Solomon. *Algorithms for the vehicle routing and scheduling problems with time window constraints*. Operations Research, vol. 35, pages 254–265, 1987. (Cité en pages 48 et 89.)

[Subramanian 2010a] A Subramanian, L M A Drummond, C Bentes, L S Ochi et R Farias. *A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery*. Computers & Operations Research, vol. 37, no. 11, pages 1899–1911, Novembre 2010. (Cité en pages 81, 82 et 90.)

[Subramanian 2010b] Anand Subramanian, Eduardo Uchoa et LuizSatoru Ochi. *New Lower Bounds for the Vehicle Routing Problem with Simultaneous Pickup and Delivery*. In Paola Festa, editeur, Experimental Algorithms SE - 24, volume 6049 of *Lecture Notes in Computer Science*, pages 276–287. Springer Berlin Heidelberg, 2010. (Cité en pages 82 et 90.)

[Subramanian 2013a] Anand Subramanian, Eduardo Uchoa et Luiz Satoru Ochi. *A hybrid algorithm for a class of vehicle routing problems*. Computers & Operations Research, vol. 40, no. 10, pages 2519–2531, 2013. (Cité en page 20.)

[Subramanian 2013b] Anand Subramanian, Eduardo Uchoa, ArturAlves Pessoa et LuizSatoru Ochi. *Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery*. Optimization Letters, vol. 7, no. 7, pages 1569–1581, 2013. (Cité en pages 58, 82 et 90.)

[Sun 2007] Lijun Sun, Xiangpei Hu, Zheng Wang et Minfang Huang. *A Knowledge-Based Model Representation and On-Line Solution Method for Dynamic Vehicle Routing Problem*. In Yong Shi, GeertDick Albada, Jack Dongarra et PeterM.A. Sloot, editeurs, Computational Science ? ICCS 2007 SE - 32, volume 4490 of *Lecture Notes in Computer Science*, pages 218–226. Springer Berlin Heidelberg, 2007. (Cité en page 16.)

[Tang 2013] Jiafu Tang, Yuyan Ma, Jing Guan et Chongjun Yan. *A Max ?Min Ant System for the split delivery weighted vehicle routing problem*. Expert Systems with Applications, vol. 40, no. 18, pages 7468–7477, 2013. (Cité en page 20.)

[Tasan 2010] A S Tasan et M Gen. *A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries*. In Computers and Industrial Engineering (CIE), 2010 40th International Conference on, pages 1–5, 2010. (Cité en page 58.)

[Tavakkoli-Moghaddam 2006] R Tavakkoli-Moghaddam, A R Saremi et M S Ziaee. *A memetic algorithm for a vehicle routing problem with backhauls*. Applied Mathematics and Computation, vol. 181, no. 2, pages 1049–1060, 2006. (Cité en page 18.)

[Toth 1999] Paolo Toth et Daniele Vigo. *A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls*. European Journal of Operational Research, vol. 113, no. 3, pages 528–543, 1999. (Cité en page 18.)

[Toth 2001] Paolo Toth et Daniele Vigo, editeurs. The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. (Cité en pages 1, 7, 10, 11 et 17.)

[Trunfio 2015] GiuseppeA. Trunfio. *Adaptation in Cooperative Coevolutionary Optimization*. In Iztok Fister et Iztok Fister Jr., editeurs, Adaptation and Hybridization in Computational Intelligence SE - 4, volume 18 of *Adaptation, Learning, and Optimization*, pages 91–109. Springer International Publishing, 2015. (Cité en page 27.)

[Tütüncü 2009] G Yazg ? Tütüncü, Carlos A C Carreto et Barrie M Baker. *A visual interactive approach to classical and mixed vehicle routing problems with backhauls*. Omega, vol. 37, no. 1, pages 138–154, 2009. (Cité en page 18.)

[Ursani 2011] Ziauddin Ursani, Daryl Essam, David Cornforth et Robert Stocker. *Localized genetic algorithm for vehicle routing problem with time windows*. Applied Soft Computing, vol. 11, no. 8, pages 5375–5390, 2011. (Cité en page 19.)

[Černý 1985a] V Černý. *Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm*. Journal of Optimization Theory and Applications, vol. 45, no. 1, pages 41–51, 1985. (Cité en page 35.)

[Černý 1985b] V Černý. *Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm*. Journal of Optimization Theory and Applications, vol. 45, no. 1, pages 41–51, 1985. (Cité en page 69.)

[Wallace 1876a] Alfred Russel Wallace. The geographical distribution of animals., volume v.1. New York,Harper and brothers,, 1876.

http ://www.biodiversitylibrary.org/bibliography/11354. (Cité en page 28.)

[Wallace 1876b] Alfred Russel Wallace. The geographical distribution of animals., volume v.2. New York,Harper and brothers,, 1876. http ://www.biodiversitylibrary.org/bibliography/11354. (Cité en page 28.)

[Wang 2007] Jiang-qing Wang, Xiao-nian Tong et Zi-mao Li. *An Improved Evolutionary Algorithm for Dynamic Vehicle Routing Problem with Time Windows*. In Yong Shi, GeertDick Albada, Jack Dongarra et PeterM.A. Sloot, editeurs, Computational Science ? ICCS 2007 SE - 171, volume 4490 of *Lecture Notes in Computer Science*, pages 1147–1154. Springer Berlin Heidelberg, 2007. (Cité en page 16.)

[Wang 2009] Fan Wang, Yi Tao et Ning Shi. *A Survey on Vehicle Routing Problem with Loading Constraints*. In Lean Yu, Kin Keung Lai et S K Mishra, editeurs, CSO (2), pages 602–606. IEEE Computer Society, 2009. (Cité en page 17.)

[Wang 2013] Chao Wang, Fu Zhao, Dong Mu et JohnW. Sutherland. *Simulated Annealing for a Vehicle Routing Problem with Simultaneous Pickup-Delivery and Time Windows*. In Vittal Prabhu, Marco Taisch et Dimitris Kiritsis, editeurs, Advances in Production Management Systems. Sustainable Production and Service Supply Chains SE - 21, volume 415 of *IFIP Advances in Information and Communication Technology*, pages 170–177. Springer Berlin Heidelberg, 2013. (Cité en page 59.)

[Wilson 1977] N H M Wilson, N J Colvin, Massachusetts Institute of Technology. Center for Transportation Studies, United States. Urban Mass Transportation Administration et Rochester Genesee Regional Transit Authority. Computer control of the Rochester dial-a-ride system. Massachusetts Institute of Technology, Center for Transportation Studies, 1977. (Cité en page 16.)

[Wolpert 1997] David H. Wolpert et William G. Macready. *No free lunch theorems for optimization*. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, vol. 1, no. 1, pages 67–82, 1997. (Cité en pages 26 et 61.)

[Yang] Xin-She Yang et Suash Deb. *Cuckoo Search via Lévy Flights*. In NaBIC, pages 210–214. IEEE. (Cité en pages 37 et 38.)

[Yu 2011] Shiwei Yu, Chang Ding et Kejun Zhu. *A hybrid GA-TS algorithm for open vehicle routing optimization of coal mines material*. Expert Systems with Applications, vol. 38, no. 8, pages 10568–10573, 2011. (Cité en page 14.)

[Zachariadis 2010] Emmanouil E Zachariadis, Christos D Tarantilis et Chris T Kiranoudis. *An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries*. European Journal of Operational Research, vol. 202, no. 2, pages 401–411, 2010. (Cité en pages 81, 82 et 90.)

[Zhan 2009] Zhi-Hui Zhan et Jun Zhang. *Discrete Particle Swarm Optimization for Multiple Destination Routing Problems*. In Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing : EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG, EvoWorkshops '09, pages 117–122, Berlin, Heidelberg, 2009. Springer-Verlag. (Cité en page 38.)

[Zhang 2008] Jing-Ling Zhang, Yanwei Zhao, Dian-Jun Peng et Wanliang Wang. *A Hybrid Quantum-Inspired Evolutionary Algorithm for Capacitated Vehicle Routing Problem.* In De-Shuang Huang, Donald C Wunsch II, Daniel S Levine et Kang-Hyun Jo, editeurs, ICIC (1), volume 5226 of *Lecture Notes in Computer Science*, pages 31–38. Springer, 2008. (Cité en page 20.)

[Zhang 2009a] Jingling Zhang, Yanwei Zhao, Dianjun Peng et Wanliang Wang. *A hybrid Quantum-Inspired Evolutionary Algorithm for open vehicle routing problem.* In Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on, pages 839–844, 2009. (Cité en pages 20 et 48.)

[Zhang 2009b] Tao Zhang, Yue-Jie Zhang, Qi Chen et Yan Sun. *The mixed algorithm for vehicle routing problem with simultaneous pick-up and delivery.* In Machine Learning and Cybernetics, 2009 International Conference on, volume 4, pages 1871–1876, 2009. (Cité en page 58.)

[Zhengchu 2010] Wang Zhengchu, Zhou Muxun, Li Xiufeng, Fan Chun et Jin Feixiang. *A quantum particle swarm optimization for solving the capacitated vehicle routing problem.* In Intelligent Control and Automation (WCICA), 2010 8th World Congress on, pages 3281–3285, 2010. (Cité en page 19.)

Résumé : L'objectif de cette thèse est de prospecter les voies de l'adaptation et de la coopération dans les métaheuristiques pour la résolution des problèmes de tournées de véhicules VRP. Nous nous intéressons à deux variantes du VRP, à savoir le problème HVRPMBTW (problème de tournées de véhicules avec flotte hétérogène, collecte et livraison et fenêtre de temps)et VRPSPD (problème de tournées de véhicules avec collecte et livraison simultanées). Le premier problème a été traité à l'aide de métaheuristiques relativement récentes, en utilisant le concept de l'adaptation qui dans notre cas consiste à régler dynamiquement les paramètres de la métaheuristique en fonction de l'instance utilisée. Le deuxième problème a été traité en utilisant la stratégie de coopération entre les métaheuristiques qui consiste à faire coopérer plusieurs métaheuristiques esclaves dont le fonctionnement est supervisé par une métaheuristique maitre dans le but de profiter des avantages offerts par chacune des métaheuristiques esclaves.

Mots clés : Problème de tournées de véhicules, Collecte et livraison, Métaheuristiques coopératives, Métaheuristiques adaptatives,Optimisation basée sur la biogéographie, Recuit simulé, La descente, La recherche coucou, La recherche harmonie, Algorithme génétique.