REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IA6/M2/2019

Mémoire

présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Intelligence artificielle**

Résolution du problème de tournées de véhicules avec contrainte du temps par l'algorithme Big Bang-Big Crunch

Par :

KHELFA AMANI

Soutenu le 07 juillet 2019, devant le jury composé de :

ALOUI Imane	M.C.B	Président
BERGHIDA Meryam	M.C.B	Rapporteur
KELFALI Toufik	M.A.A	Examinateur

Tout d'abord, je remercie le Dieu, de nous avoir donné la force, la volonté et le courage afin d'accomplir ce travail modeste.

J'adresse le grand remerciement à ma encadrant Mme ***BERGHIDA Meryem*** qui a dirigé ce travail, pour ses conseils sa disponibilité, sa gentillesse et son attention durant toute la période du travail.

Mes vifs remerciements vont également aux membres de jury : à Mme. ***ALOUI Imane*** et M. ***KELFALI Toufik*** pour l'honneur qu'ils j'ai fait en portant leur attention sur ce travail.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je désire aussi remercier les enseignants de département de l'informatique de l'université Mohamed Khider Biskra, qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

Je voudrais exprimer ma reconnaissance envers les amis(es) et collègues qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche.

Finalement, je tiens à exprimer notre profonde gratitude à ma famille qui par leurs prières et leurs encouragements, on a pu surmonter tous les obstacles.

Dédicace

Je dédie ce mémoire a :

A Mes très chers parents que Dieu les garde et les protège pour tout ce qu'ils ont fait pour moi.

A ma tendre Mère : Tu représentes pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie.

A mon très cher Père : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être.

A mon encadreur Mme **BERGHIDA Meryem**. Merci mille fois pour votre gentillesse, votre patience, votre disponibilité, votre compétence et votre compréhension.

A mes sœurs **Asma**, **Assila** et **Anfel** mes sources de bonheur. Les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous. Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

A mon cher frère : **Mohammed Amine**, j'espère que la vie lui réserve le meilleur.

A tous mes oncles, mes tantes, mes cousins et mes cousines pour toute l'affection qu'ils m'ont donnée et pour leurs précieux encouragements.

A mes très chers amies **Djihene**, **Sabrine**, **Rima** et **Roufaida**, mes sœurs, vous m'avez toujours soutenue, m'encouragé. Aucune dédicace ne saurait exprimer l'amour, l'estime que j'ai toujours pour vous, je remercie Dieu de t'avoir connue.

A mes collègues de l'option Intelligence artificielle pour leurs encouragements et leurs précieuses aides. A tous les étudiants de ma promotion.

Tous ce qui ont enseigné moi au long de ma vie scolaire.

A tous les étudiants de ma promotion.

Tous ceux qui, par un mot, m'ont donné la force de continuer …..

Merci à vous tous.

Résumé

L'objectif de ce mémoire est la résolution des problèmes de tournées de véhicules VRP utilisant les métaheuristiques. Nous nous intéressons à une variante du VRP, à savoir le problème VRPTW (problème de tournées de véhicules avec fenêtre de temps). Ce problème a été traité à l'aide d'une métaheuristiques appelée l'algorithme Big Bung-Big Crunch une nouvelle méthode d'optimisation basée sur l'une des théories de l'évolution de l'univers, à savoir la théorie du Big Bang et du Big Crunch. On a amélioré cet algorithme par l'utilisation de deux méthodes ou opérateurs heuristiques d'amélioration, à savoir la méthode 2-opt et la méthode 1-1 exchange.

Mots clés : Problème de tournées de véhicules, Fenêtre de temps, Algorithme Big Bung-Big Crunch, Recherche locale, Opérateur heuristique d'amélioration, Méthode 2-opt, Méthode 1-1 exchange.

Abstract

The objective of this thesis is the resolution of VRP vehicle touring problems using metaheuristics. We are interested in a variant of the VRP, namely the problem VRPTW (Vehicle Routing Problem with Time Windows). This problem was treated using a metaheuristics called the Big Bung-Big Crunch algorithm a new optimization method that is based on one of the theories of the evolution of the universe namely the Big Bang and Big Crunch theory. This algorithm has been improved by the use of two heuristic improvement methods or operators, namely the 2-opt method and the 1-1 exchange method.

Keywords: Vehicle routing problem, Time Windows, Big Bung-Big Crunch algorithm, Local Search, Heuristic improvement operator, 2-opt method, 1-1 exchange method.

Liste des tableaux

Table des figures

Table des matières

Introduction générale

Les enjeux économiques et environnementaux renforcent l'importance de l'optimisation combinatoire dans les domaines d'informatique et la recherche opérationnelle. C'est pour cela que de nos jours, l'optimisation des transports et la performance logistique sont des enjeux économiques très importants pour les entreprises parce que les problèmes de transport jouent un rôle très important dans de nombreux secteurs d'activités. Ce type de problèmes est rencontré, par exemple, dans les chaînes logistiques de grands groupes industriels, dans la grande distribution, dans les systèmes de transports publics (métro, bus,...) ou privés. Les problèmes se rencontrent aussi bien dans le cadre des services à la personne (transport scolaire, transport des personnes âgées,...) que dans le cadre du transport de marchandises (livraison de colis,...).

On peut considérer que résoudre un problème de transport consiste à organiser dans le temps et dans l'espace un ensemble de tournées affectées à divers véhicules. Ainsi, la plupart des solutions commerciales proposant une optimisation des tournées n'aboutissent pas souvent sur de véritables outils d'optimisation mais proposent plutôt, des solutions organisationnelles. Ces solutions sont souvent exploitées pour rationaliser les coûts des livraisons, des distributions ou encore des collectes.

Les problèmes de tournées de véhicules, ou Vehicle Routing Problem (VRP dans la suite de ce mémoire), figurent parmi les problèmes d'optimisation combinatoire les plus étudiés. Le VRP modélise une situation où une flotte de véhicules, se trouvant dans un dépôt, doit servir un ensemble de clients géographiquement dispersés. Le but est alors de déterminer quels clients doit visiter chaque véhicule et dans quel ordre de sorte que l'activité soit effectuée le plus efficacement possible.Ce problème étant classé NP-difficile [1], sa résolution nécessite l'utilisation des métaheuristiques.

Les principales motivations de l'étude du VRP sont d'une part la difficulté de sa résolution et d'autre part ses nombreuses applications pratiques en logistique. Ce deuxième point concerne les retombées économiques et environnementales liées à la minimisation des coûts des systèmes de transport.

De nombreuses variantes du problème ont été étudiées. Nous nous intéresserons aux problèmes de tournées de véhicules avec fenêtres de temps (VRPTW pour Vehicle Routing Problem with Time Windows).

Pour le VRPTW, le service à chaque clients doit être effectué à l'intérieur d'un

intervalle de temps, appelé fenêtre de temps. Celui-ci peut être rigide (hard) ou flexible (soft). Dans le premier cas, si le véhicule arrive avant l'heure permise, il doit attendre cette heure pour débuter le service, sans pénalité ; par contre, il n'est pas permis de dépasser la limite supérieure de la fenêtre de temps. Lorsque l'intervalle de temps est flexible, le véhicule peut arrive avant la borne inférieure ou après la borne supérieure. Dans ce cas, il existe deux versions du problème. Dans la première, le véhicule arrivant avant la borne inférieure doit attendre pour débuter le service, alors que s'il arrive en retard, il peut effectuer leur service moyennant un cout supplémentaire (pénalité de retard). Pour ce qui est de la second version, le véhicule peut débuter le service à l'extérieure des limites permises par la fenêtre de temps, au prés de certain pénalité.

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. 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 donc aucun algorithme efficace pour les grandes instances n'est connu pour la résolution de la plupart de ces problèmes. Mais les approches heuristiques de résolution qui ont été développées, visent à fournir de bonnes solutions dans un délai raisonnable pour un problème donné. Néanmoins, ces méthodes ont deux inconvénients majeurs lesquels :

• Elles sont adaptées à un problème spécifique, et que leur adaptation à d'autres problèmes est difficile et souvent même impossible.

• 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. Récemment, de nombreuses techniques d'optimisation basées sur la population ont été utilisées pour résoudre des problèmes d'optimisation complexes, comme les algorithmes génétiques, les algorithmes de colonies de fourmis, l'évolution différentiel, l'optimisation basée sur la biogéographie. . . etc.

Dans ce mémoire, nous nous intéressons à une métaheuristique à base de population qui est l'algorithme BB-BC [2] une nouvelle méthode d'optimisation basée sur l'une des théories de l'évolution de l'univers, à savoir la théorie du Big Bang et du Big Crunch.

Le présent document est structuré de la façon suivante :

Nous introduisons ce mémoire avec une introduction générale.

Le premier chapitre présente la problématique du VRP. Dans ce chapitre, nous rappelons les différents éléments qui composent ce problème, ainsi que les contraintes

à satisfaire et l'objectif à optimiser. La formulation mathématique de ce problème est ensuite rappelée. Nous rappelons quelques variantes de ce problème trouvées dans la littérature, puis nous présentons un état de l'art sur les méthodes qui sont utilisées pour la résolution de ce problème.

Le deuxième chapitre présente l'algorithme Big Bung-Big Crunch. Dans ce chapitre, le big bang – big crunch (BB – BC), une méthode d'optimisation globale inspirée de l'une des théories cosmologiques connues sous le nom d'un univers fermé, est présenté. Nous décrivons d'abord les deux théories Big Bang-Big Crunch.Ensuite, Nous présentons l'algorithme générale BB-BC, son pseudo code, sa description détaillée et quelques travaux tirés de la littérature. Enfin, tire la conclusion de ce chapitre.

Notre contribution est présentée dans le troisième et le quatrième chapitre.

Le troisième chapitre présente la phase d'adaptation de l'algorithme BB-BC pour résoudre le problème VRPTW. Dans ce chapitre nous décrivons la manière d'encodage de la solution, ensuite nous donnons le pseudo code de l'approche utilisée pour résoudre le VRPTW avec une description détaillée de ce dernier.

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

Nous concluons ce mémoire avec une conclusion générale.

Chapitre 1

Problème de tournées de véhicules

1.1 Introduction

Dans l'économie actuelle, rare est le produit qui arrive à être consommé par son utilisateur final sans l'intervention du transport. Presque tous les produits doivent passer par une série de déplacements entre un lieu de production quelconque, des dépôts et des consommateurs.

Les enjeux économiques et environnementaux renforcent l'importance de l'optimisation combinatoire dans les domaines d'informatique et la recherche opérationnelle. C'est pour cela que de nos jours, l'optimisation des transports et la performance logistique sont des enjeux économiques très importants pour les entreprises. Cela repose notamment sur l'organisation et l'efficacité des tournées de véhicules. Ainsi, la plupart des solutions commerciales proposant une optimisation des tournées n'aboutissent pas souvent sur de véritables outils d'optimisation mais proposent plutôt, des solutions organisationnelles. Ces solutions sont souvent exploitées pour rationaliser les coûts des livraisons, des distributions ou encore des collectes. Le problème VRP (pour Vehicle Routing Problem), bien que simple à énoncer, est NP-difficile [1], tous les problèmes de cette classe se ramènent à celui-ci via une réduction polynomiale avec l'utilisation des méthodes approchées.

1.2 Le problème de tournées de véhicules VRP

Le problème de tournées de véhicules (Vehicle Routing Problem) est une généralisation du TSP obtenue lorsque K véhicules de capacité Q sont disponibles au nœud-dépôt. Chaque véhicule doit donc effectuer une tournée réalisable, c'est à dire quitter le dépôt, visité une fois des clients dont la somme des demandes ne dépasse pas la capacité Q, avant de retourner au dépôt. Le but est de trouver des tournées de cout minimal permettant de livrer tous les clients et respectant les contraintes de capacité

des véhicules.[3]

Le VRP implique la planification de routes de livraison à moindre coût, afin de servir un ensemble de clients dispersés géographiquement, tout en respectant les contraintes de capacité des véhicules. Il existe différentes extensions des problèmes de tournées prenant en compte des objectifs et contraintes opérationnelles des contextes d'application visés.

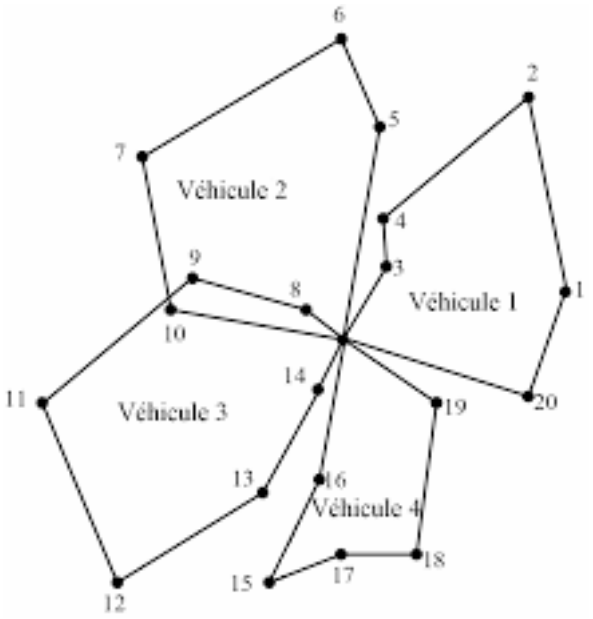

FIGURE 1.1 – Un exemple de problème de VRP 20 clients résolu avec 4 véhicules [4]

Il existe de nombreuses formulations de problème de tournée de véhicule.
La formulation suivante est tirée de FISHER et JAI KUMAR (1981).

Paramètres

n = Nombre de nœuds.
k = Nombre de véhicules.
cap = Capacité d'un véhicule.
T^k = Temps maximal de tournée du véhicule k.
d_i = Demande du nœud i.
t_i^k = Temps nécessaire au véhicule k pour charger ou décharger au nœud i.
t_{ij}^k = Temps nécessaire au véhicule k pour voyager du nœud i au nœud j.
C_{ij} = Coût ou distance du voyage du nœud i au nœud j.

Variables de décision

x_{ij}^k = { 1 si le véhicule k voyage du nœud i vers le nœud j.

La fonction objectif

Minimiser $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} C_{ij} * x_{ij}^k$

Les contraintes de problème

$$\sum_{i=0}^{n}\sum_{k=1}^{m} x_{i\,j}^{k} = 1 \qquad j = 0, \cdots, n \qquad (1.1)$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{i\,j}^{k} = 1 \qquad i = 0, \cdots, n \qquad (1.2)$$

$$\sum_{i=0}^{n} x_{i\,p}^{k} - \sum_{j=0}^{n} x_{pj}^{k} = 0 \qquad k = 1, \cdots, m; p = 1, \cdots, n \qquad (1.3)$$

$$\sum_{i=0}^{n} t_{i}^{k} \sum_{j=0}^{n} x_{i\,j}^{k} + \sum_{i=0}^{n}\sum_{j=0}^{n} t_{i\,j}^{k} * x_{i\,j}^{k} \leq T^{k} \qquad k = 1, \cdots, m \qquad (1.4)$$

$$\sum_{i=0}^{n} d_{i} \sum_{j=0}^{n} x_{i\,j}^{k} \leq cap \qquad k = 1, \cdots, m \qquad (1.5)$$

Les équations (1.1) et (1.2) assurent que chaque nœud n'est servi qu'une seule fois par un et un seul véhicule.
L'équation (1.3) assure la continuité d'une tournée par un véhicule : le nœud visité doit impérativement être quitté.
L'équation (1.4) assure le respect de la contrainte de la durée totale d'une tournée.
L'équation (1.5) assure le respect de la contrainte de capacité du véhicule.

1.3 Quelques variantes du VRP

1.3.1 CVRP (Capacitated Vehicle Routing Problem)

Les véhicules ont une capacité d'emport limitée (quantité, volume, poids).
Données : Un ensemble de nœuds client (ayant une demande) et d'arêtes (munies de coûts) et une flotte illimitée de véhicules de capacité uniforme Q partant d'un unique dépôt.
Trouver : Des tournées de véhicules, satisfaisant chaque demande une et une seule fois et respectant les contraintes de capacité.
Minimisant : Le coût total de transport lié aux arcs empruntés par les véhicules et/ou les Coûts fixes associés à l'utilisation des véhicules.[5]

$$\sum_{i \in \mathcal{N}} q_{i} y_{i}^{k} \leq \mathbf{Q} \qquad \forall k \in \mathbf{K} \qquad (1.6)$$

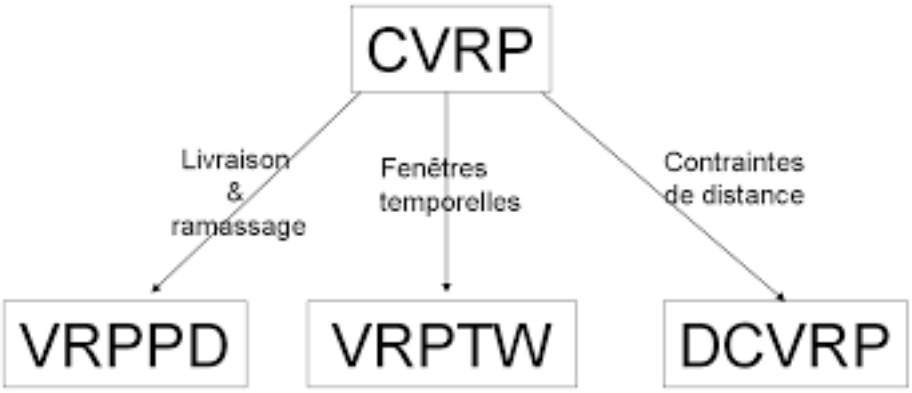

FIGURE 1.2 – Variantes de base du VRP avec contraintes de capacité [4]

1.3.2 VRPTW (Vehicle Routing Problem with Time Windows)

Le VRP avec fenêtre de temps désigne l'ensemble des problèmes de tournées de véhicules pour lesquels les interventions sont soumises à des fenêtres de temps. Ces problèmes sont fréquemment rencontrés dans la vie réelle, dans lesquels les contraintes de type « rendez-vous » (visite en fonction de la disponibilité du client) sont très présentes.

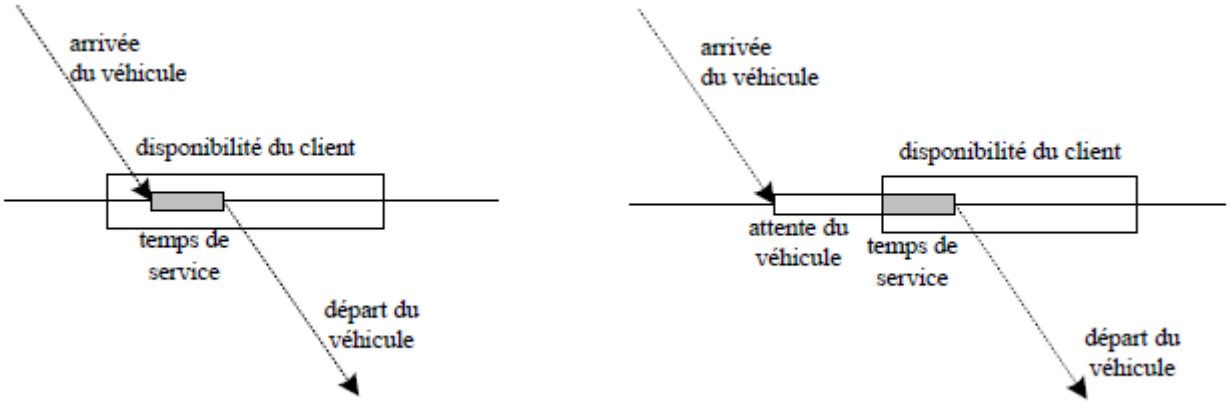

FIGURE 1.3 – Exemple de fenêtre de temps sur un nœud

Données : Un ensemble de nœuds client ayant une demande et des fenêtres de temps (souples ou dures) ; un ensemble d'arêtes (munies de coûts) et une flotte illimitée de véhicules de capacité fixée **Q** partant d'un unique dépôt.
Trouver : Un ensemble des tournées de véhicules, satisfaisant chaque demande une et une seule fois en respectant les contraintes de capacité des véhicules et les contraintes de fenêtres de temps dures.
Minimisant : Le coût total de transport lié aux arcs empruntés par les véhicules et/ou le coût lié au non-respect des fenêtres de temps souples.
Il est possible d'étendre le modèle de Fisher et Jaikumar pour tenir compte des fenêtres

de temps.[5]
Pour cela, nous avons besoin de données et de variables temporelles. Nous définissons :
- [ai,bi] la fenêtre de temps au nœud i. - s_i le temps de service au nœud i.
- t_{ij} le temps de transport entre les nœuds i et j.
- la variable u_i^k associée à l'heure d'arrivée du véhicule k au nœud i.
- **M** une grande valeur.
Nous allons présenter les contraintes traitant les fenêtres de temps :

$a_i \leq u_i^k \leq b_i \qquad \forall i \in \mathcal{N}, \forall k \in \mathbf{K}$

$u_i^k + s_i + t_{ij} - \mathbf{M} * (1 - x_{ij}^k) \leq u_j^k \qquad \forall i \in \mathcal{N}, \forall j \neq 1 \in \mathcal{N}, \forall k \in \mathbf{K}$

1.3.3 VRPHF (Vehicle Routing Problem with Heterogeneous Fleet)

La flotte est composée de véhicules de types différents, qui se distinguent par la capacité, la puissance, le coût de transport,... [6]

1.3.4 OVRP (Open Vehicle Routing Problem)

Le véhicule est libre de rejoindre ou pas le dépôt après la fin de la tournée. S'il choisit de reprendre le dépôt, il doit reprendre le parcours de la tournée dans le sens inverse.[6]

1.3.5 VRPMD (Multi-Depot Vehicle Routing Problem)

Les véhicules peuvent s'approvisionner de plusieurs dépôts.[6]

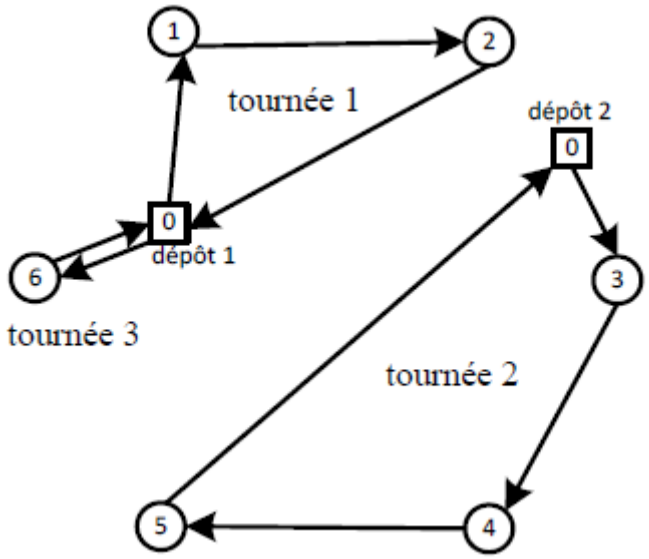

FIGURE 1.4 – Exemple de solution multi-dépôt d'un VRP

1.3.6 1-VRP (Problème de Tournées de Véhicule à un Seul Produit)

Un seul produit doit être livré aux différents clients par chaque véhicule en une seule tournée.[6]

1.3.7 VRPPD (Vehicle Routing Problem with Pick-up and Delivery)

Les véhicules effectuent un double service : la livraison (Delivery) des clients ainsi que le ramassage (Pick-up) de marchandises de ces derniers. Le ramassage induit une difficulté supplémentaire en rapport avec la contrainte de capacité des véhicules. Les quantités de marchandises à ramasser sont notées pi $1 \leq i \leq n$ et par convention, lorsqu'un véhicule dessert un client, il commence par la livraison puis effectue le ramassage et par convention, lorsqu'un véhicule dessert un client il commence par la livraison puis exécute le ramassage. Le dépôt est par défaut le lieu de départ des marchandises à livrer et le lieu des marchandises ramassées.[4]

1.4 Les méthodes de résolution de VRP et ses variantes

Les méthodes de résolution des problèmes d'optimisation comme les problèmes de transport tel que VRP sont généralement divisées en deux catégories les méthodes exactes et les méthodes approchées (heuristiques et métaheuristiques).

1.4.1 Les méthodes exactes

Les méthodes exactes, appelées aussi méthodes complètes, permettent de trouver la solution optimale d'un problème d'optimisation en explorant exhaustivement l'ensemble des solutions (ou configurations) possibles. L'exploration énumérative (toutes les solutions sont évaluées une à une) est la technique la plus basique mais elle reste inappropriée aux problèmes combinatoires.[4]

La plupart des problèmes de transport peuvent être présentés sous forme d'un PLNE (Programmation Linéaire en Nombres Entiers), dans le but de minimiser (ou maximiser) une fonction linéaire à variables entières en respectant un ensemble de contraintes linéaires.

En novembre 2016, Mohammad Mirabi et al ont présenté une méthode exacte pour résoudre le problème de routage de véhicule multi-dépôt avec une fenêtre de temps dans lequel le véhicule n'est pas nécessaire de retourner à son dépôt principal après avoir

servi les clients. La méthode est comparée à deux autres techniques de regroupement, les algorithmes de moyennes floues (FCM) et les algorithmes K-means. Les résultats expérimentaux montrent la robustesse et l'efficacité de l'algorithme proposé.[7]

Dans 2011, Andrea Bettinelli et al a 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.[8]

1.4.2 Les heuristiques

Par définition, une heuristique est un moyen de guider les choix que doit faire un algorithme pour réduire sa complexité. Une heuristique est spécifique à un problème et ne peut pas être généralisée.[4]

L'heuristique de Clarke et Wright qui construit les tournées à partir d'un seul client pour tous les autres clients, puis, elle fusionne ces tournées deux par deux, tant que cela permet de réduire le coût de la solution courante et que chaque nouvelle tournée respecte les contraintes du problème. Dans cette heuristique, on commence par fusionner les tournées qui réduisent le plus de coûts. Cette heuristique présente un double avantage, celui de réduire à la fois le nombre de véhicules utilisés et le coût total. C'est pourquoi, elle est efficace sur la plupart des instances.[9]

En octobre 2016, Sohaib Afifi et al. ont présenté une méthode heuristique basé sur recuit simulé (SA-FILS) pour résoudre le problème de routage de véhicule avec fenêtres temporelles et visites synchronisées. Ils montrent que leur méthode est rapide et surpasse les approches existantes, et elle produit toutes les solutions optimales connues de la référence dans des temps de calcul très courts et améliore les meilleurs résultats pour le reste des instances.[10]

1.4.3 Les métaheuristiques

Les métaheuristiques peuvent être vues comme des heuristiques « puissantes et évoluées » dans la mesure où elles sont généralisables à plusieurs problèmes d'optimisation. Les métaheuristiques sont habituellement classées en fonction du nombre de solutions qu'elles manipulent : les métaheuristiques à solution unique (méthodes à trajectoire) telles que la recherche tabou [Fred W. Glover en 1986] ; le recuit simulé [Kirkpatrick et al. en 1983]...etc. Et les métaheuristiques à base de population telles que les algorithmes génétiques [holland 1970] et les colonies de fourmis [Marco Dorigo et al. 1990] $\cdots$ etc.

Les métaheuristiques appliquées aux problèmes de tournées de véhicules ont reçu beaucoup d'attention de la communauté scientifique.

En novembre 2018, Alfian Fai et al. ont présenté des métaheuristiques pour résoudre le problème de routage de véhicule par capacité (CVRP). Cette approche combinait la

recherche de voisinage variable (PVNS) basée sur la perturbation avec le mécanisme de sélection adaptative (ASM) pour contrôler le schéma de perturbation. Les résultats informatiques prouvent que la méthode développée est compétitive et très efficace pour obtenir une solution de haute qualité dans des délais de calcul raisonnables.[11]

1.5 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.

Dans ce chapitre, nous avons présenté le problème de tournées de véhicules (le problème VRP), et décrit les différentes extensions de ce problème tel que VRPTW, VRPMD et OVRP. Nous avons introduit les méthodes de résolution utilisées pour ce problème comme les méthodes exactes, les heuristiques et les métaheuristiques et nous avons donné des exemples des travaux déjà fait pour résoudre un VRP en utilisant une de ces méthodes.

Dans le chapitre suivant, nous présentons une métaheuristique qu'est l'algorithme Big-Bang Big-Crunch utilisé pour résoudre le problème VRPTW.

Chapitre 2

L'algorithme Big-Bang Big- Crunch

2.1 Introduction

La théorie cosmologique est un sujet passionnant, car elle montre comment l'univers se produit, se déplace et se révolte. L'un des sujets fascinants est la provenance de toutes les étoiles et galaxies. Cette question a longtemps été explorée par la physique. Par exemple, deux physiciens célèbres, Isaac Newton et Albert Einstein, ont estimé que l'univers ne change pas et ont introduit un terme appelé constante cosmologique[12]. En 1929, l'astronome Edwin Hubble a découvert que l'univers était en expansion. Il a découvert qu'au début de l'univers, la gravité était forte, du fait de la concentration de la matière dans un très petit espace, qu'elle était comprimée en un seul point[13]. Ainsi, il subirait une pression incroyable et s'est élargi depuis, connu sous le nom de big bang. Cet événement a été controversé jusqu'en 1965, quand une découverte accidentelle a soutenu la théorie. Aujourd'hui, les observations astronomiques les plus avancées montrent que la théorie du big bang est probablement vraie.

2.1.1 Big Bang

Littéralement, chaque morceau de matière et d'énergie de notre univers a été créé par le biais d'un événement singulier (c'est-à-dire l'inimaginable creuset de chaleur et de lumière) que nous appelons le big bang. Cet événement s'est passé il y a $13 * 10^9$ ans. Bien que cette théorie ne soit pas parfaite et que, au fil du temps, la physique ait fait des efforts pour la rendre plus cohérente. Une chose est vraie, l'univers était en train de devenir ce que nous observons aujourd'hui : rempli de galaxies, d'étoiles, de planètes et de toutes sortes de choses étranges et exotiques.

2.1.2 Big Crunch

Une hypothèse selon laquelle l'avenir de l'univers s'appelle modèle de «big crunch», c'est-à-dire que l'univers se contracte en un point de masse. Cela se déroulera presque exactement comme le big bang, sauf en sens inverse. Que l'expansion de l'univers ait lieu pour toujours ou s'arrête un jour, cela dépend de la quantité de matière qu'il contient.[14]

2.2 L'algorithme Big Bang-Big Crunch

2.2.1 Principes fondamentaux de l'algorithme Big Bang - Big Crunch

Inspiré de l'une des théories cosmologiques voulant que l'univers soit «né» dans le big bang et puisse «mourir» dans le big-crunch. Erol et Eksin [2] ont proposé un nouvel algorithme, à savoir le «Big Bang - Big Crunch». (BB – BC) algorithme. En général, l'algorithme proposé comprend deux phases successives.

La phase Big Bang

Tout comme l'expansion de l'univers, l'objectif principal de la phase big bang de BB – BC est de créer des populations initiales. La position initiale de chaque entrée est générée aléatoirement sur tout l'espace de recherche. Une fois le pool de population créé, les valeurs de condition physique des individus sont calculées.[15]

La phase Big Crunch

Si suffisamment de masse et d'énergie (c.-à-d. Entrées) se trouvent dans l'univers (c.-à-d. L'espace de recherche), cette masse et cette énergie peuvent alors générer suffisamment d'attrait pour interrompre l'expansion de l'univers et l'inverser, ramenant ainsi l'univers entier à un point unique. De la même manière, dans la phase critique, une procédure de contraction est appliquée pour former un centre ou un point représentatif pour les opérations Big Bang ultérieures. En d'autres termes, la phase critique est un opérateur de convergence qui a plusieurs entrées, mais une seule sortie, appelée "centre de masse". Ici, le terme masse fait référence à l'inverse de la valeur de la fonction de fitness(f^i).

2.2.2 Pseudo code de l'algorithme Big Bang-Big Crunch

Phase Big Bang (Construction de solutions)

Etape 1 : Générer une population (construire des solutions à partir de zéro pour la

première génération, ou bien générer une nouvelle population du pool d'élite).

Phase Big Crunch (Déplacement de la recherche locale)

Répéter

Etape 2 : Générer des voisins pour toutes les solutions de la population et remplacer le parent par son meilleur descendant pour chaque solution dans la population.

Etape 3 : Trouver le centre de masse.

Etape 4 : Appliquer la recherche locale au centre de masse.

Etape 5 : Mettre à jour le pool élite et la meilleure solution trouvée.

Etape 6 : Éliminer certaines solutions de mauvaise qualité.

Jusqu'à ce que la taille de la population soit réduite à une solution unique.

Etape 7 : Retourner à l'étape 1 si le critère d'arrêt n'est pas rempli.

Etape 8 : Renvoyer la meilleure solution trouvée.[16]

2.2.3 Description de l'algorithme Big Bang-Big Crunch

D'abord, l'algorithme BB-BC commence par la phase de construction (Big Bang) qui construit une population de N solutions candidates possibles à partir de zéro *(étape 1)* pour la première génération. Pour la phase successive du Big Bang, on génère une nouvelle population à partir du pool d'élite.

Ensuite, nous utilisons la phase de Big Crunch (amélioration), où on génère d'abord quelques voisins de toutes les solutions de la population, en effectuant des perturbations simples. Chaque solution (parent) sera remplacée par la meilleure descendant afin d'avoir des solutions de meilleure qualité dans une population successive *(étape 2)*. À l'*(étape 3)*, on détermine le centre de masse en fonction de la meilleure valeur de coût de solution trouvée. Le centre de masse est encore amélioré en appliquant une heuristique de descente simple pour un nombre prédéfini d'itérations sans amélioration *(étape 4)*. Un pool d'élite (collection) est créé / mis à jour à l'*(étape 5)*, où les meilleures solutions (centre de masse) des générations précédentes sont stockées dans le pool d'élite et seront exploitées comme solutions de référence pour la phase du Big Bang lors des itérations successives. À chaque itération, le pool d'élite est mis à jour en remplaçant le coût de la solution le plus défavorable dans le centre de masse actuel et les solutions, la recherche convergera progressivement vers une solution unique en réduisant la taille de la population *(étape 6)*. Cela se fait en éliminant les solutions de mauvaise qualité autour du centre de masse. La phase Big Crunch est répétée jusqu'à ce que la taille de la population soit réduite à une solution unique (singularité). Un nouveau Big Bang commence *(étape 7)* en revenant à la première étape, où une nouvelle population est générée à partir du pool d'élite. C'est-à-dire incluant les solutions d'élite dans la nouvelle population et en générant des voisins pour former la nouvelle population.

Enfin, le BB-BC renvoie la meilleure solution trouvée *(étape 8)*.[16]

2.3 Performance de l'algorithme Big Bang-Big Crunch

Pour évaluer les performances de l'algorithme BB–BC, Erol et Eksin[2] ont proposé six fonctions de test, à savoir la fonction Sphere, la fonction de Rosenbrock, la fonction Step, la fonction Ellipsoïde, la fonction de Rastrigin et la fonction Ackley. Comparé à l'algorithme génétique de combat (CGA), l'algorithme BB–BC présentait de meilleurs résultats pour la recherche de la meilleure solution globale.

2.4 Quelques variantes de l'algorithme Big Bang-Big Crunch

Bien que l'algorithme BB – BC soit un nouveau membre de la famille de l'intelligence computationnelle (IC), un certain nombre de variations de BB – BC ont été proposées dans la littérature afin d'améliorer encore les performances de BB – BC. Cette section donne un aperçu de certaines de ces variantes de BB – BC qui se sont révélées très efficaces et robustes.

2.4.1 L'algorithme Big Bang - Big Crunch hybride

Dans Kaveh et Talatahari[17] les auteurs ont mis au point l'un des premiers hybrides BB–BC, appelé hybride BB–BC (HBB–BC). Dans l'ensemble, HBB–BC a introduit deux améliorations : utilisation des capacités d'optimisation d'essaims de particules (PSO) pour améliorer la capacité d'exploration de l'algorithme BB–BC et utilisation du mécanisme de sous-optimisation (SOM) pour mettre à jour l'espace de recherche de l'algorithme BB – BC.Comparé au BB–BC standard et à d'autres méthodes d'optimisation classiques telles que l'algorithme génétique (GA), l'optimisation des colonies de fourmis (ACO), la PSO et la recherche harmonique (HS), le HBB–BC est meilleur.

2.4.2 L'algorithme Big Bang - Big Crunch amélioré

Pour améliorer les performances de BB – BC, Hasançebi et Azad[18] ont proposé deux variantes améliorées de l'algorithme BB – BC, appelées respectivement algorithmes BB – BC modifiés (MBB – BC) et BB – BC exponentielles (EBB – BC). Dans la nouvelle formulation, le nombre aléatoire normal (r) est modifié en utilisant toute distribution statistique appropriée afin d'éliminer les inconvénients de la formulation standard (par exemple, la grande dimensionnalité de recherche). En outre, pour satisfaire aux exigences de données discrètes, l'algorithme amélioré BB – BC a utilisé la méthode de l'arrondi au lieu des valeurs réelles aux entiers les plus proches représentant le numéro de séquence.

2.4.3 L'algorithme Big Bang-Big Crunch pour résoudre le problème du fonctionnement du réservoir

Dans le compte rendu de la 12ème conférence internationale sur la qualité de la recherche (QiR), Mohammad Hadi Afshar et Isa Motaei[19] ont montré dans leur papier de conférence que l'algorithme Big Bang-Big Crunch (BB-BC), est utilisé pour résoudre le problème du fonctionnement du réservoir. Le but de cette étude était d'évaluer la performance de l'algorithme dans la résolution du problème de fonctionnement du réservoir. Pour cela, le problème de l'alimentation en eau et de l'exploitation hydroélectrique du réservoir de Dez en Iran sur une période de 5 à 20 ans est présenté comme une étude de cas ; les résultats sont présentés et comparés à ceux obtenus par l'algorithme MMAS (Max-Min Ant System).Le résultat confirme la capacité de l'algorithme BB-BC à résoudre de manière optimale le problème du fonctionnement du réservoir.

2.4.4 L'algorithme Big Bang-Big Crunch pour les problèmes binaires

Dans le journal IJISAE, Ismail KOC[20] a publié son étude dans laquelle la méthode BB-BC est modifiée pour traiter les problèmes d'optimisation binaire. La performance des méthodes proposées est analysée sur les problèmes de localisation d'installations non condensés (PFUU), qui sont l'un des problèmes binaires utilisés dans la littérature. Les douze et les petites instances bien connues d'UFLP sont utilisés pour analyser les performances et les effets du paramètre de contrôle de l'algorithme BB-BC. Les résultats obtenus sont présentés de manière comparative. Selon les résultats expérimentaux, la version binaire de la méthode BB-BC donne de bons résultats dans la résolution de problèmes liés aux UFLP en termes de qualité de la solution.

2.5 Conclusion

Dans ce chapitre, nous avons présenté l'algorithme big bang–big crunch (BB–BC). Nous décrivons d'abord les deux théories Big Bang-Big Crunch.Ensuite, Nous présentons l'algorithme générale BB-BC, son pseudo code, sa description détaillée et quelques travaux tirés de la littérature.

Dans le chapitre suivant, on fait l'adaptation de l'algorithme BB-BC pour résoudre le problème VRPTW.

Chapitre 3

Résolution du problème VRPTW avec l'algorithme BB-BC

3.1 Introduction

Dans les chapitres précédents nous avons présenté le problème de tournée de véhicule et l'algorithme Big Bang-Big Crunch.

Dans ce chapitre nous présentons l'adaptation proposée de cet algorithme pour la résolution du problème VRPTW (Vehicle Routing Problem with Time Windows).

3.2 Encodage de la solution

3.2.1 Définition de la structure de données

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 tournées. Chaque tournée est représentée par un véhicule. Chaque élément du vecteur est une liste de clients visités par le véhicule, en commençant par le dépôt et en terminant dans le dépôt.

Exemple Supposant qu'on ait 5 clients (codés de 1 à 5) et 4 véhicules. 0 représente le nœud de dépôt. Une solution faisable est représentée dans la table 3.1. Cette solution utilise seulement trois véhicules, la première visite le client 3, le client 1 ensuite il retourne au dépôt. Le deuxième visite le client 2 ensuite 4 et retourne au dépôt alors que le troisième seulement le client 5. Le dernier est non utilisé.

T1	T2	T3
0	0	0
3	2	5
1	4	0
0	0	

TABLE 3.1 – Exemple d'une solution faisable

3.2.2 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 une tournée tout en respectant les contraintes du problème. Dans le problème VRPTW, la capacité est limitée et le client avait un temps prédéfinit pour être servis.

Pour générer une solution faisable aléatoirement, nous commençons une tournée, et nous affectons des clients aléatoires à cette tournée en respectant les contraintes de temps et de capacité. Une fois la capacité du véhicule soit insuffisante ou pas de clients approprié, nous commençant une autre tournée et ainsi de suite jusqu'à ce que tous les clients soient servis.

3.2.3 Méthode 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, mais nous utilisons un seul type. Nous présentons dans ce qui suit ce type.

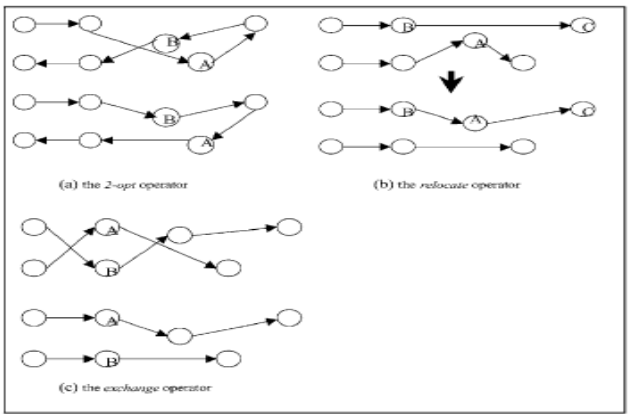

FIGURE 3.1 – Opérateurs heuristiques d'amélioration utilisés par Tam and Ma 2004 pour résoudre un CVRPTW [21]

Calcul de voisinage par permutation de clients d'une même tournée

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 la solution originel.

Dans la méthode 2-opt, les voisins sont calculés par permutation des clients au sein d'une même tournée, ceci revient à modifier l'ordre de visite des clients par le véhicule. Dans ce cas la contrainte de capacité reste vérifié et on vérifié la contrainte de temps.

Exemple Supposons que l'exemple représenté dans la table 3.1 soit une solution candidate pour la mutation. Supposons que la tournée aléatoirement sélectionnés pour être échangés dans le processus de mutation soit T2. La table 3.2 représente la tournée après la mutation.

T1	T2	T3
0	0	0
3	4	5
1	2	0
0	0	

TABLE 3.2 – Tournée après la mutation

Algorithme Mouvement de permutation de clients d'une même tournée

---

Choisir aléatoirement une tournée Ti d'une solution S ;
Choisir aléatoirement deux clients c1,c2 dans Ti ;
Permuter c1 et c2 ;
si verfier_solution(S) alors
    Retourner vrai ;
Retourner faux ;

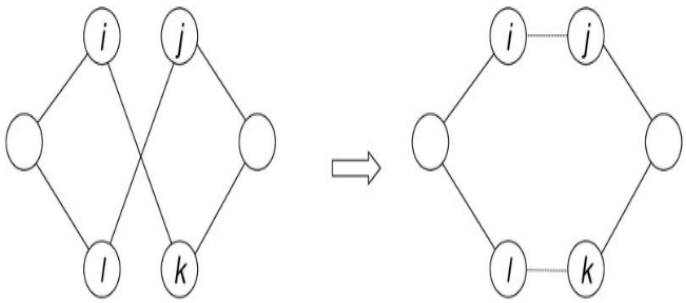

FIGURE 3.2 – Principe 2-opt [22]

3.2.4 Méthode de la recherche locale

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 les contraintes de problème (temps, capacité).

Algorithme Mouvement de permutation de clients entre deux tournées

---

Choisir aléatoirement deus tournées Ti, Tj d'une solution S ;
Choisir aléatoirement deux clients c1, c2 dans Ti,Tj ;
Permuter c1 et c2 ;
si verfier_solution(S) alors
    Retourner vrai ;
Retourner faux ;

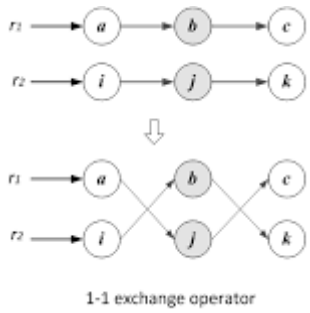

FIGURE 3.3 – Principe 1-1 exchange [23]

3.2.5 Rectification d'une solution

La génération des voisinages par la mutation ne garantit pas la faisabilité de la solution soit en termes de capacité ou en termes de temps.

Affin d'éviter des temps exorbitants dans la génération des voisinages faisable, 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, et les affecter dans des nouvelles tournées.

3.3 BB-BC pour le VRPTW : pseudo code

Phase Big Bang (Construction de solutions)

Etape 1 : Générer une population aléatoire en vérifiant les contraintes de problème.

Phase Big Crunch (Déplacement de la recherche locale)

Répéter

Etape 2 : Générer des voisins pour toutes les solutions de la population en utilisant la méthode 2-opt et remplacer le parent par son meilleur descendant pour chaque solution dans la population.

Etape 3 : Trouver le centre de masse. (Minimum coût)

Etape 4 : Appliquer la recherche locale en utilisant la méthode 1-1 exchange.

Etape 5 : Mettre à jour le pool élite et la meilleure solution trouvée.

Etape 6 : Éliminer certaines solutions de mauvaise qualité.

Jusqu'à ce que la taille de la population soit réduite à une solution unique.

Etape 7 : Retourner à l'étape 1 si le critère d'arrêt n'est pas rempli. (Nombre itérations)

Etape 8 : Renvoyer la meilleure solution trouvée.

3.4 Description de l'algorithme

D'abord, nous commençons l'algorithme BB-BC par la phase de construction (Big Bang) qui construit une population de N solutions candidates possibles en vérifiant les contraintes de problème(temps, capacité) *(étape 1)*.

Ensuite, nous utilisons la phase de Big Crunch (amélioration), où on génère d'abord quelques voisins de toutes les solutions de la population en utilisant la méthode 2-opt. Chaque solution (parent) sera remplacée par le meilleur descendant *(étape 2)*. À l'*(étape 3)*, on détermine le centre de masse en fonction de la meilleure valeur de coût de solution trouvée. Le centre de masse est encore amélioré en appliquant la méthode 1-1 exchange *(étape 4)*. Un pool d'élite (collection) est créé / mis à jour à l'*(étape 5)*, où les meilleures solutions (centre de masse) des générations précédentes sont stockées dans le pool d'élite et seront exploitées comme solutions de référence pour la phase du Big Bang lors des itérations suivantes. À chaque itération, le pool d'élite est mis à jour en remplaçant le coût de la solution le plus défavorable (solution ayant le coût minimal) dans le centre de masse actuel et les solutions, la recherche convergera progressivement vers une solution unique en réduisant la taille de la population *(étape 6)*. Cela se fait en éliminant les solutions de mauvaise qualité autour du centre de masse. La phase Big Crunch est répétée jusqu'à ce que la taille de la population soit réduite à une solution unique. Un nouveau Big Bang commence *(étape 7)* en revenant à la première étape, où une nouvelle population est générée à partir du pool d'élite. C'est-à-dire incluant les solutions d'élite dans la nouvelle population et en générant des voisins pour former la nouvelle population. Enfin, le BB-BC renvoie la meilleure solution trouvée *(étape 8)*.[16]

3.5 Conclusion

Dans ce chapitre nous avons présenté l'adaptation de l'algorithme BB-BC pour résoudre le problème VRPTW. Nous avons décrit la manière d'encodage de la solution, ensuite nous avons donné le pseudo code de l'approche utilisée pour résoudre le VRPTW avec une description détaillée de cette dernière.

Chapitre 4

Résultats et Expérimentation

4.1 Introduction

Ce chapitre est consacré à la présentation de l'environnement de développement utilisé pour réaliser notre projet, ainsi que les données de tests, les résultats obtenus, comparaison et discussion de ces résultats.

4.2 Environnement de développement

Pour achever notre travail il faut savoir tout d'abord que les étapes de réalisation de notre travail reposent sur deux éléments : l'environnement matériel et l'environnement logiciel.

4.2.1 Environnement matériel

Durent le développement de notre approche, on a utilisé un ordinateur «hp» possédant les caractéristiques suivantes :

- Modèle : hp ProBook 4530s
- Processeur : Intel(R) Core(TM) i5-2430M CPU @ 2.4 GHz 2.40 GHz
- Mémoire : 4GØ
- Systéme d'exploitation : Windows7 Edition Intégrale, 32 bits

4.2.2 Environnement logiciel

La réalisation de notre projet n'exige pas logiciel spécifique. Nous utilisons eclipse.

Eclipse est un environnement de développement (IDE) historiquement destiné au langage Java, même si grâce à un système de plugins il peut également être utilisé avec d'autres langages de programmation, dont le C/C++ et le PHP. Eclipse nécessite une

machine virtuelle Java (JRE) pour fonctionner. Mais pour compiler du code Java, un kit de développement (JDK) est indispensable.

FIGURE 4.1 – Eclipse icône

4.2.3 Langage de programmation

Le langage de programmation qu'on a choisi pour le développement de notre approche est le langage java.

Java, est un langage de programmation orienté objet utilisable sur divers systèmes d'exploitation, il est assez robuste, portable et à hautes performances.

FIGURE 4.2 – Java icône

4.3 Représentation de l'application

Pour simplifier l'utilisation de notre approche, on a crée une interface graphique simple qui donne la main à l'utilisateur d'introduire les paramètres de l'approche proposée. Pour cela, nous utilisons la bibliothèque Swing sous eclipse.

Swing est une bibliothèque graphique pour le langage de programmation Java, elle offre la possibilité de créer des interfaces graphiques.

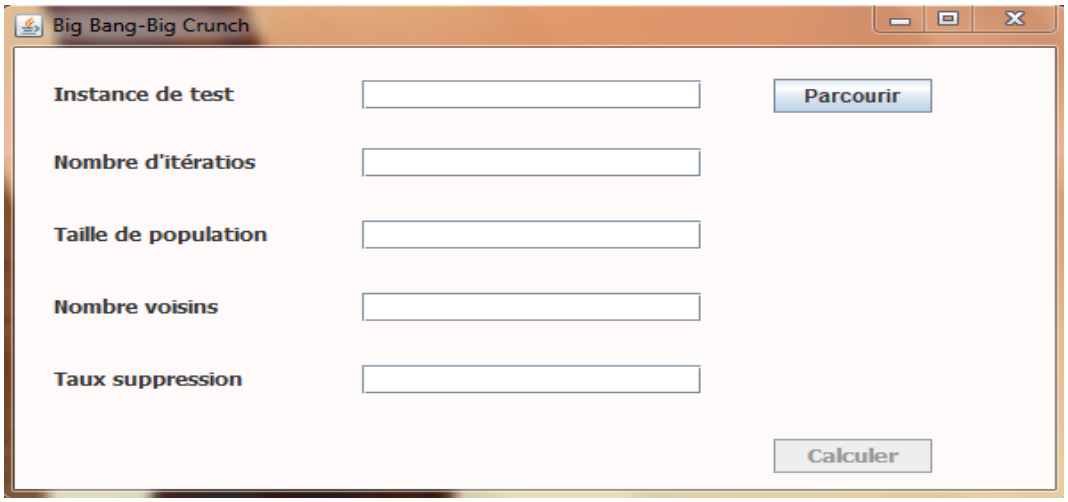

FIGURE 4.3 – Notre application(bouton Calculer non affiché)

Le bouton **Parcourir** pour choisir l'instance de test, le bouton **Calculer** reste inactif jusqu'à ce que tous les champs soient remplis.

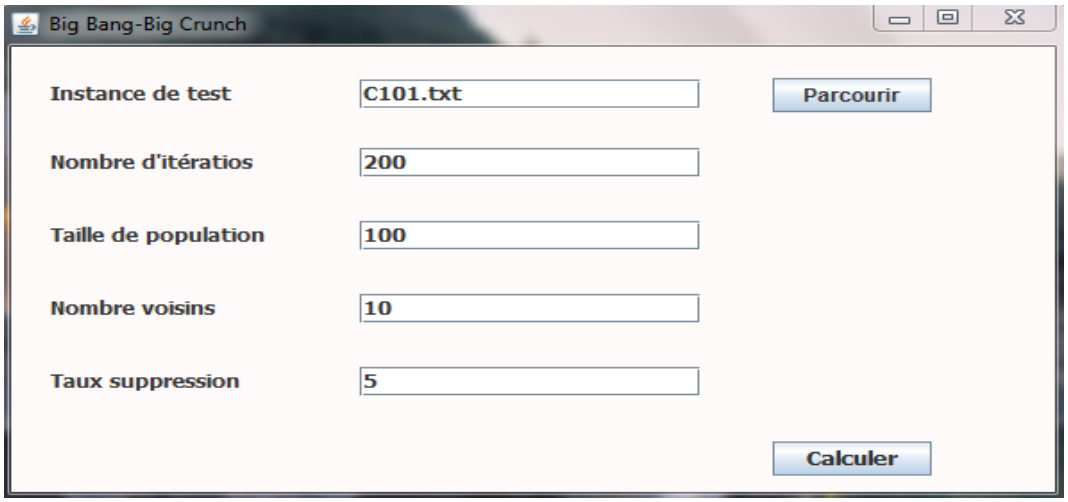

FIGURE 4.4 – Notre application(bouton Calculer affiché)

Les résultats (coût, temps d'exécution) sont affichés dans une autre interface. Pour calculer le temps d'exécution on utilise l'équation suivante

TempExecution = TempsFin - TempsDebut telque :

TempsDebut :temps avant de commencer l'exécution.

TempsFin : temps de fin d'exécution.

Le bouton **Retour** pour commancer un nouveau calcul.

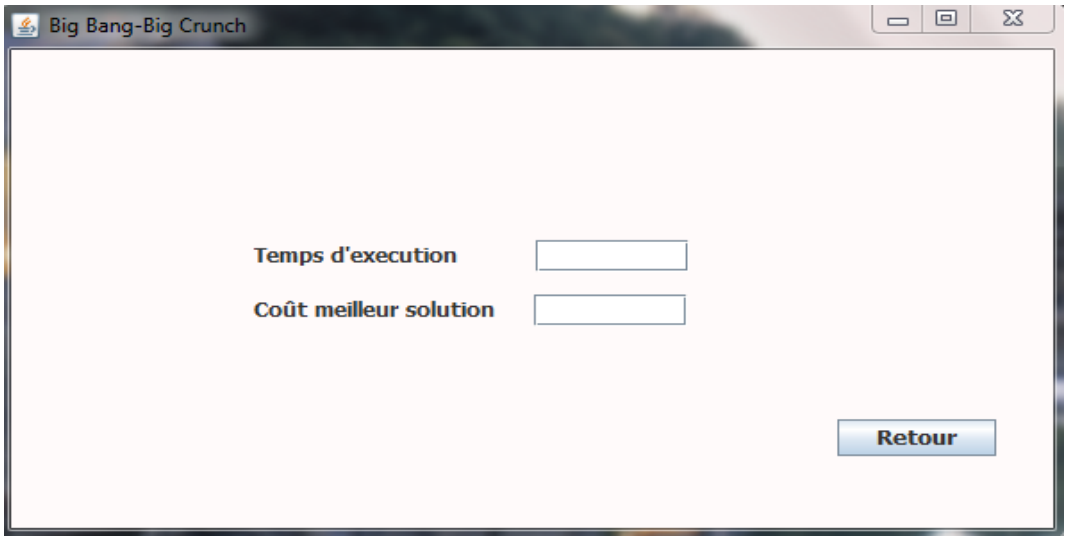

FIGURE 4.5 – Notre application(afficher résultats)

4.4 Données de tests

Dans ce travail, nous avons utilisé des données existantes pour ce problème. Nous utilisons comme base les instances de Solomon pour le problème de VRPTW (Vehicle Routing Problem with Time Windows) [23] avec 25 clients, 50 clients et 100 clients. Nous conservons les données concernant chaque client : id, coordonnées, demande, fenêtre de visite, temps de service. Nous générons ensuite, une matrice de distance en utilisant la distance euclidienne.

C101

VEHICLE

NUMBER	CAPACITY
25	200

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	914	965	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	144	90
25	25	52	40	169	224	90

FIGURE 4.6 – Exemple d'instance de Solomon avec 25 clients

4.5 Résultats, comparaison et discussion

4.5.1 Jeux de données utilisé

Nous testons notre approche sur différents ensembles de problème dérivés de l'ensemble de données de Solomon[23] 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 nœuds. 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.

4.5.2 Comparaison de notre approche avec une méthode exacte

Nous commençons les expériences en comparant les résultats de notre approche BB-BC avec les résultats obtenus par une méthode exacte pour les instances de petite taille (5 clients). Les résultats exacts sont trouvés en utilisant LPSolve IDE. Le tableau 4.1 montre que notre approche proposée donne des résultats proches de l'optimal (Nb représente le nombre d'instances, ME représente les résultats obtenus par la méthode exacte). Cela est dû au fait que l'espace de recherche est réduit.

Instance	Nb	ME	BB-BC
C1-5	9	77.45	77.85
C2-5	8	110.19	110.19
R1-5	12	152.66	153.20
R2-5	11	145.19	145.19

TABLE 4.1 – Solution moyenne de 5 clients

4.5.3 Comparaison de BB-BC avec VNS-C

Dans cette section, nous comparons l'approche proposée avec VNS-C (Variable Neighbourhood Search-Compound)[24] à l'aide de l'opérateur GAP qui calcul l'écart entre deux solution, et qui est défini comme suit :

$$GAP_{approche1-approche2} = \frac{Cost_{approche1} - Cost_{approche2}}{Cost_{approche2}} \times 100\% \qquad (4.1)$$

Les résultats sont résumés dans les tableaux 4.2, 4.3 et 4.4. La première colonne représente le nom de l'instance, les résultats obtenus par VNS-C sont présentés dans la deuxième et les résultats de la approche proposée BB-BC.

A partir des tableaux 4.2, 4.3 et 4.4 nous remarquons que :
- Dans le premier type d'instances (C1, C2), où les clients sont regroupés, les résultats obtenus étaient proches de ceux de l'approche VNS-C, mais notre approche n'a pas les amélioré.
- Dans le deuxième type d'instances (R1, R2), où les clients sont distribués normalement, BB-BC donnait des meilleurs résultats, il a amélioré les résultats de l'approche VNS-C avec une moyenne de 0.58%.
- Dans le dernier type d'instances (RC1, RC2), notre approche a amélioré les résultats de 4 instances (RC101, RC102, RC104, RC204) et dans 2 instances elle a donné les même résultats (RC201, RC206) et dans le reste (8) la différence était petite.

Instance	VNS-C	BB-BC	GAP %
C101-100	828.94	828.94	0
C102-100	828.94	828.94	0
C103-100	828.94	831.79	-0.34
C104-100	825.64	823.94	0.20
C105-100	828.94	828.94	0
C106-100	828.94	831.79	-0.34
C107-100	828.94	836.25	-0.87
Moyenne			-0.19
C201-100	591.56	588.88	0.46
C202-100	591.56	591.56	0
C203-100	591.17	599.16	-1.3
C204-100	590.60	590.60	0
C205-100	588.88	588.88	0
C206-100	588.49	588.49	0
C207-100	588.29	588.29	0
Moyenne			-0.42
Moyenne C1, C2			-0.30

TABLE 4.2 – Comparaison de 100 clients (C1, C2) pour l'approche BB-BC

Instance	VNS-C	BB-BC	GAP %
R102-100	1476.06	1486.12	-0.68
R103-100	1219.89	1219.89	0
R104-100	994.85	974.24	2.1
R105-100	1381.88	1377.11	0.35
R106-100	1243.72	1234.6	0.74
R107-100	1077.24	1051.84	2.41
Moyenne			0.82
R202-100	1098.06	1091.21	0.63
R203-100	905.72	905.72	0
R204-100	766.91	789.72	-2.89
R205-100	964.02	954.16	1.03
R206-100	931.76	935.04	-0.35
R207-100	855.37	825.52	3.62
Moyenne			0.34
Moyenne R1, R2			0.58

TABLE 4.3 – Comparaison de 100 clients (R1, R2) pour l'approche BB-BC

Instance	VNS-C	BB-BC	GAP %
RC101-100	1624.97	1619.8	0.32
RC102-100	1467.25	1466.84	0.03
RC103-100	1265.86	1284.24	-1.43
RC104-100	1136.49	1171.61	0.35
RC105-100	1642.81	1629.44	-2.3
RC106-100	1396.59	1408.7	-0.86
RC107-100	1254.68	1258.32	-0.29
Moyenne			-0.6
RC201-100	1310.44	1310.44	0
RC202-100	1219.49	1233.46	-1.13
RC203-100	957.10	1020.72	-6.23
RC204-100	829.13	798.46	3.84
RC205-100	1233.46	1273.03	-3.11
RC206-100	1107.40	1107.40	0
RC207-100	1032.78	1061.14	-2.67
Moyenne			-1.33
Moyenne R1, R2			-0.97

TABLE 4.4 – Comparaison de 100 clients (RC1, RC2) pour l'approche BB-BC

4.6 Conclusion

Dans ce chapitre nous avons présenté l'environnement de développement ainsi que le langage de programmation utilisé. Nous présentons par la suite les données de tests utilisées, les résultats obtenus lors de l'application de notre approche et enfin la discussion de ces résultats.

Conclusion générale

Cette partie conclut notre mémoire en rappelant les objectifs et les points abordés. Le travail présenté dans ce mémoire traite du 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 constituent une grande famille de problèmes d'optimisation combinatoire avec des applications dans de nombreux domaines différents, allant de la distribution de marchandises à la fourniture de services. Dans ce mémoire, nous avons étudié une variante du problème de tournées de véhicules, à savoir, le problème de tournées de véhicules avec fenêtres de temps (VRPTW pour Vehicle Routing Problem with Time Windows). L'objectif était d'adapter une métaheuristique basée sur l'une des théories de l'évolution de l'univers appelée l'algorithme Big Bung-Big Crunch pour résoudre le problème cité.

La méthode d'optimisation Big-Bang Big-Crunch, comme son nom l'indique, la méthode repose sur l'application continue de deux étapes successives, à savoir les phases Big Bang et Big Crunch. Dans la phase du Big Bang, certaines solutions possibles au problème d'optimisation sont générées de manière aléatoire et réparties sur l'espace de recherche. Lors de la phase Big Crunch, les solutions candidates distribuées de manière aléatoire sont dessinées dans un seul point représentatif via une approche de centre de population ou de coût minimal.

Perspective

Comme perspectives, nous aimerions pouvoir améliorer l'approche proposée en la combinant avec d'autres méthodes comme le recuit simulé et l'utiliser comme un opérateur d'amélioration de chaque meilleure solution dans chaque itération.

Bibliographie

[1] G Laporte. “The vehicle routing problem : An overview of exact and approximate algorithms”. In : *European Journal of Operational Research* 59.3 (1992), 345–358 (Cité en pages 2 et 17).

[2] Osman K.Erol et Ibrahim Eksin. “A new optimization method : Big Bang–Big Crunch”. In : *International Journal of Advances in Intelligent Informatics* 37.2 (2006), p. 106–111.

[3] GUIBADJ Rym Nesrine. *Problèmes de tournées de véhicules et application industrielle pour la réduction de l'empreinte écologique*. France, 2013.

[4] BEN ISMAIL Sahbi et LEGRAS François et COPPIN Gilles. “Synthèse du problème de routage de véhicules”. Rapport. 2011.

[5] MALAPERT Arnaud. *Optimisation de tournées de véhicules pour l'exploitation de Réseau Telecom*. France, 2006.

[6] MERZAK Zoulikha et ABBAZ Souheyla. *Problème de tournées de véhicules avec gestion de stock dans un réseau de distribution*. Algerie, 2016.

[7] SHOKRI Nasibeh et SADEGHIEH Ahmad MIRABI Mohammad. “Modeling and Solving the Multi-depot Vehicle Routing Problem with Time Window by Considering the Flexible end Depot in Each Route”. In : *International Journal of Supply and Operations Management (IJSOM)* 3.3 (2016), p. 1373–1390.

[8] Alberto Ceselli et Righini Giovanni Albert EinsteinAndrea Bettinelli. “A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows”. In : *Transportation Research* 19.5 (2011), p. 723–740.

[9] G Clarke et JW Wright. “Scheduling of vehicles from a central depot to a number of delivery points”. In : *Operations research* 12.4 (1964), p. 568–581.

[10] DC Dang et A Moukrim S Afifi. “Heuristic solutions for the vehicle routing problem with time windows and synchronized visits”. In : *Transportation Research* 10.5 (2016), p. 511–525.

[11] Subiyanto Subiyanto et Ulfah Mediaty Arief Alfian Faiz. "An efficient meta-heuristic algorithm for solving capacitated vehicle routing problem". In : *International Journal of Advances in Intelligent Informatics* 4.3 (2018).

[12] Tamara Davis et Brendan Griffen. *Cosmological constant*. 2010. URL : `http://www.scholarpedia.org/article/Cosmological_constant`.

[13] *DE L'ASTRONOMIE : Edwin HUBBLE, le découvreur*. 2010. URL : `http://cepheides.fr/article-de-l-astronomie-edwin-hubble-le-decouvreur-49930693.html`.

[14] John Scalzi. *The rough guide to the universe*. Reading, Massachusetts : Rough Guides, 2008.

[15] İbrahim Eksin et Osman K. Erol Hakki M. Genç. "Big Bang - Big Crunch optimization algorithm hybridized with local directional moves and application to target motion analysis problem". In : 2010, p. 1448–1452.

[16] Ghaith M. Jaradat et Masri Ayob. "Big bang-big crunch optimization algorithm to solve the course timetabling problem". In : 2010, p. 1448–1452.

[17] A.Kaveh et S.Talatahari. "Size optimization of space trusses using Big Bang–Big Crunch algorithm". In : *Computers and Structures* 87.17-18 (2009), p. 1129–1140.

[18] O.Hasançebi et S.Kazemzadeh Azad. "An exponential big bang–big crunch algorithm for discrete design optimization of steel frames". In : *Computers and Structures* 110-111 (2012), p. 167–179.

[19] Mohammad Hadi Afshar et Isa Motaei. "Big Bang-Big Crunch Algorithm : Application to Reservoir Operation problems". In : 2011, p. 2294–2299.

[20] Ismail KOC. "Big Bang-Big Crunch Optimization Algorithm for Solving the Uncapacitated Facility Location Problem". In : *International Journal of Intelligent Systems and Applications in Engineering* (2016), p. 185–189.

[21] Tam et K.T. Ma. "Combining meta-heuristics to effectively solve the vehicle routing problems with time windows". In : *Artificial Intelligence Review* 21 (2004), p. 87–112.

[22] Berghida Meryem. *Stratégies adaptatives et coopératives pour la résolution de problèmes de tournées de véhicules avec collecte et livraison*. Algerie, 2015.

[23] Marius M Solomon. "Algorithms for the vehicle routing and scheduling problems with time window constraints". In : *Operations Research* 35 (1987), p. 254–265.

[24] Ruibin Bai et Hisao Ishibuchi Rong Qu Binhui Chen. "A Variable Neighbourhood Search Algorithm with Compound Neighbourhoods for VRPTW". In : 2016, p. 25–35.