République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed Seddik Ben Yahia –Jijel-
Faculté des Sciences Exactes et Informatique
Département de l'Informatique

Mémoire
De Fin d'Etude pour l'obtention du Diplôme de
Master en Informatique

Option : Intelligence Artificielle

Thème

Approche Adaptative pour la résolution du Problème de Tournées de Véhicules avec Fenêtres de Temps

Réalisé par:
Boussaid adlan
Fanit Salih

Encadré par :
Dr. Meryem Berghida

جامعة محمد الصديق بن يحي - جيجل
كلية العلوم
و الإعلام الآلي

Promotion 2016

M. ing. IA 01/16

Remerciement :

*Nous remercions avant tout, « **Allah** » le tout puissant qui nous a éclairé le chemin de la réussite et du savoir et nous a donné le courage et la volonté pour réaliser ce travail.*

*Nous remercions tout d'abord notre directrice de mémoire **Dr Meryem Berghida** qui a accepté de nous encadrer et consacrer son temps à nous orienter afin d'accomplir ce travail.*

Nous tenons à remercier également les membres du jury, pour avoir accepté de juger ce mémoire.

Nous remercions tous les enseignants qui ont été à la base de notre formation, et nos collègues de la promotion 2016.

DEDICACE

Après cette réussite que fait la joie
A tous qui m'aime. Je dédie ce modeste travail
avec vif plaisir à ceux qui sont les plus proches à
mon cœur, qui mon toujours aimer et qui ont fait
de toute leurs force pour que je sois toujours
heureux. mes très chères parents, qui sans eux je
serai rien ;
À mes frères ;
À toute ma grande famille ;
À mes amies.

Salih et Adlan.

Table des Matières

Introduction Générale 1

Chapitre 1 : Optimisation combinatoire

1.1 Introduction 5

1.2 Notions de base 6

1.2.1 Diversification.....6

1.2.2 Intensification et Apprentissage 6

1.2.3 Voisinage 6

1.2.4 Optimum local et Optimum global 6

1.2.4.1 Optimum local 6

1.2.4.2 Optimum global 6

1.3 La compléxité 7

1.3.1 Compléxité d'algorithme.....7

1.3.2 Compléxité de probléme 8

1.4 Méthodes de résolution des problèmes d'optimisation.....8

1.4.1 Méthodes exactes 9

1.4.2 Heuristiques.....9

1.4.3 Métaheuristiques.....9

1.5 Conclusion 12

Chapitre 2 : Problèmes de tournées de véhicules

2.1 Introduction 13

2.2 Le Probléme de tournées de véhicules.....13

2.3 Variantes de VRP 14

2.3.1 CVRP (Capacitated Vehicle Routing Problem).....15

2.3.2 VRPTW (Vehicle Routing Problem with Time Windows).....17

2.3.3 VRPB (Vehicle Routing Problem with Backhauls).....17

2.3.4 VRPPD (Vehicle Routing Problem with Pickup and Delivery).....18

2.3.5 OVRP (Open Vehicle Routing Problem) 19

2.3.6 HVRP (Heterogeneous Vehicle Routing Problem).....20

2.4 Les méthodes de résolution de VRP et ses variantes 21

2.4.1 Travaux sur l'état de l'art 21
2.4.2 Les méthodes exactes 22
2.4.3 Les méthodes hybrides 22
2.5 Conclusion 23

Chapitre 3 : Approche adaptative pour la résolution de VRPTW

3.1 Introduction 24
3.2 Modélisation de VRPTW 24
3.3 Technique d'adaptation.....25
3.4 Résolution de VRPTW avec l'algorithme de chauve-souris.....27
3.4.1 BA : Bat Algorithm(algorithme de chauve-souris).....27
3.4.2 EDBA (algorithme amélioré de chauve-souris discret pour la résolution de VRPTW).....28
3.4.2.1 Répresentation de position et vitesse.....29
3.4.2.2 Equation de mise a jour de position.....29
3.4.2.3 La recherche de voisinage (recherche locale) 30
3.4.3 Algorithme EDBA 30
3.4.3.1 Encodage de solution.....31
3.4.4 Description de l'algorithme 32
3.5 Conclusion 39

Chapitre 4 : Résultats et expérimentation

4.1 Introduction 40
4.2 Environnement de programmation.....40
4.3 Choix du langage 40
4.4 Les jeux de données 41
4.5 Présentation de l'interface 44
4.6 Résultats 47
4.6.1 Comparaison résultats obtenus par « EDBA » avec les solutions optimales pour les instances de petites tailles.....48

4.6.2 Comparaison résultats obtenus par « EDBA » avec les meilleurs résultats dans la littérature pour les instances de grande taille.....50

4.7 Conclusion 51

Conclusion générale 53

Bibliographiques 55

Liste des tableaux

3.1- Exemple d'une solution faisable 32
4.1-Résultats obtenus par l'approche EDBA pour les instances de 25 clients 48
4.2- Résultats obtenus par l'approche EDBA pour les instances de 50 clients 49
4.3-Solutions optimaux pour les instances de 50 et 25 clients 49
4.4-Tableau EDBA Vs Solutions optimaux 49
4.5- Résultats obtenus par l'approche EDBA pour les instances de 100 clients 50
4.6-Meilleurs résultats obtenus dans la littérature pour les instances de 100 clients 51
4.7-Tableau EDBA Vs Meilleurs résultats dans la littérature(BKS) 51

Liste des figures

1.1-Optima locaux et optima globaux d'une fonction à une variable 7
1.2-Classification des méthodes de résolution de problèmes d'optimisation 12
2.1-Exemple d'un problème de type VRP.....14
2.2- Variantes du VRP.....22
3.1-Exemple de la méthode 2-échange pour VRPTW.....33
3.2-Principe 2-Opt interne d'une tournée.....35
3.3-Principe 2-Opt externe entre deux tournées.....36
3.4-Principe de permutation interne d'une tournée 37
3.5-Principe de permutation externe entre deux tournées.....37
4.1-L'organisation textuelle d'une instance de solution 42
4.2-Visualisation de trois instances (C101, R101, RC101) des problèmes de Solomon..44
4.3-L'interface de l'application 44
4.4-La fenêtre de réglage des paramètres de l'algorithme EDBA.....45
4.5-Chargement d'une instance de problème 45
4.6-L'interface après chargement d'une solution d'une instance 46
4.7-Exemple d'une solution d'une instance 47

Liste des algorithmes

3.1-Algorithme de réglage des paramètres (la descente) d'une métaheuristique M avec *n* Paramètres 26
3.2-Algorithme BAT.....28
3.3- Algorithme (EDBA).....30
3.4 -Création de la population initiale.....32
3.5-méthode 2-échange.....34
3.6-Méthode de 2-Opt interne.....35
3.7-Méthode de 2-Opt externe.....36
3.8-Mouvement de permutation de clients dans une même tournée.....37
3.9-Mouvement de permutation de clients entre deux tournées.....38
3.10-Recuit simulé.....39

Introduction générale

Le terme logistique vient du mot grec "LOGISTIKOS", qui signifie « relatif au raisonnement logique ». Au départ, il était exclusif au glossaire militaire et actuellement le mot logistique est largement utilisé en économie. En effet, une chaine logistique est un ensemble d'opérateurs garantissant la bonne circulation de marchandises, de l'approvisionnement en matière première à la commercialisation du produit fini prêt à la consommation, passant par le processus de production et les différentes étapes de stockage et de transport. Par conséquent, la logistique comprend l'ensemble des processus liés à la bonne circulation des flux physiques et administratifs, afin de satisfaire les exigences des clients en termes de qualité, coût et particulièrement en termes de temps de livraison, quelle que soit la situation géographique.

L'activité de transport est le cœur même de la logistique. L'organisation logistique est souvent déterminée par l'optimisation des coûts de transports.

De nos jours, le problème de transport de marchandises occupe une place importante dans la vie économique des sociétés modernes. Avec les contraintes temporelles et économiques qu'impliquent ce problème, sa résolution devient très difficile, nécessitant l'utilisation d'outils issus de disciplines différentes (productique, informatique, optimisation combinatoire…etc.). En effet, les processus issus des systèmes de transports et de l'ordonnancement sont de plus en plus complexes, par leurs dimensions importantes, par la nature de leurs relations dynamiques, et par la multiplicité des contraintes auxquelles ils sont soumis.

L' évolution du marché mondial, caractérisée par la diversification des sources d'approvisionnement, la spécialisation des sites de production, les échanges inter-usines et parfois même internationaux, la création de sites de stockage intermédiaires et la commercialisation des produits, fait apparaitre de nouveaux problèmes auxquels le manager doit faire face. En outre, les débats politiques sur la protection de l'environnement et la pollution par les émissions des gaz à effet de serre (CO, CO2.) et le pétrole qui se fait de plus en plus rare ainsi que l'augmentation des prix de revient des énergies non renouvelables ne fait que renforcer l'inquiétude des managers. Plus de 30% des dépenses d'une chaine logistique sont consacrées au transport, ce taux ne fera qu'augmenter avec le temps.

Le problème de transport est célèbre en optimisation combinatoire. Le problème de base et probablement le plus étudié, est le problème du Voyageur de Commerce (PVC) dit aussi Traveling Salesman Problem (TSP), qui a pour objectif, la visite aux moindres coûts d'un ensemble de clients, une et une seule fois, avec un seul véhicule.

La difficulté de ce problème est de trouver l'ordre dans lequel chacun des clients sera visité, en minimisant un certain critère (temps ou coût du parcours, ou bien longueur totale parcourue…etc.).

L'extension du nombre de véhicules d'un à une flotte de véhicules définit un autre problème de transport nommé Problème de Tournées de véhicules ou Vehicle Routing Problem (VRP). Plusieurs problèmes réels peuvent être modélisés par des VRP, à savoir la conception des systèmes de l'industrie, le traitement d'images, les problèmes rencontrés dans les réseaux industriels, la conception d'emplois du temps, le routage dans les réseaux informatiques…etc. La majorité de ces problèmes sont qualifiés difficiles, car il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable.

Plusieurs recherches se sont principalement orientées vers la résolution du VRP. Ce dernier s'agit d'un problème d'optimisation de tournées de véhicules devant satisfaire des demandes de transport. Ces problèmes sont en général sujets à plusieurs types de contraintes. Les plus étudiées sont : les contraintes de capacité et les contraintes de temps.

Les contraintes de capacité : elles sont amenées par des opérations de chargement /déchargement devant les sites et la limite de la capacité du véhicule qui doit transporter les produits.

Les contraintes de temps : apparaissent lorsque le service ne peut avoir lieu que dans un intervalle de temps donné, **fenêtre de temps** ou bien, lorsque les véhicules sont disponibles uniquement durant des périodes de temps données.

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 ce travail à la résolution approchée d'un problème de tournées de véhicules classique, auquel nous rajoutons les contraintes de temps. Le problème ainsi défini est noté VRPTW. De nombreux problèmes du processus de transport se modélisent sous la forme de VRPTW, nous citons : la distribution et ou collecte de produits agro-alimentaires périssables tels que le lait et ses dérivés, le pain, les fruits et légumes, les viandes, l'eau potable, le ramassage du personnel ou scolaire...etc. Le problème de ramassage des déchets (ménagers, hospitaliers ou industriels pour traitement et ou recyclage) représente un marché florissant ces dernières années, c'est un exemple très concret. D'autres applications du VRPTW, tels que les problèmes de tournées de maintenance (système d'alarme), de visites de médecins ou infirmiers.

L'objectif de ce mémoire est de prospecter la voie de l'adaptation dans la résolution du problème de tournées de véhicules avec fenêtre de temps (VRPTW). Ce problème a été abordé à l'aide d'une métaheuristique relativement récente nommé algorithme de chauves-souris (Bat Algorithm BA), en utilisant le concept de l'adaptation dans le but d'évaluer le comportement de cette nouvelle métaheuristique pour ce problème. 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.

Ce mémoire est constitué de quatre chapitres. Les deux premiers sont consacrés aux études théoriques. Le troisième chapitre décrit l'approche proposée et le quatrième présente les résultats d'expérimentations sur différentes instances prises de la littérature. Une brève description de ces chapitres est faite dans les paragraphes suivants :

Dans le premier chapitre, nous présentons les concepts de base de l'optimisation combinatoire. Ensuite, nous présentons les différentes méthodes de résolution de problèmes d'optimisation. Le chapitre 2 fait l'objet d'une description de problème de tournées de

véhicule VRP avec ses principales variantes ainsi que quelques méthodes de résolution présentes dans la littérature. Dans le chapitre 3, nous développons l'approche adaptative proposée pour résoudre le problème VRPTW, pour cela, nous présentons l'approche proposée, à savoir, l'approche discrète de chauves-souris améliorée. Le chapitre 4 est réservé à l'implémentation de la méthode proposée (environnement de programmation, jeux de données utilisé, résultats et comparaison avec d'autres méthodes). Ce mémoire est clôturé par une conclusion générale et perspectives.

Chapitre 1

Optimisation combinatoire

1.1 Introduction

L'optimisation combinatoire est un outil indispensable combinant diverses techniques des mathématiques et de l'informatique afin de résoudre des problèmes de la vie réelle. D'une manière simple, résoudre un problème d'optimisation combinatoire consiste à trouver l'optimum d'une fonction parmi un nombre fini de choix, souvent très grand. Il s'agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objectif sous certaines contraintes. Le but est de trouver une solution optimale dans un temps d'exécution raisonnable. Néanmoins, ce but est loin d'être concrétisé pour plusieurs problèmes vus leurs complexités grandissantes. La théorie de la NP-complétude a permis de classer les problèmes d'optimisation selon leurs complexités et a fourni des informations pertinentes sur le genre de méthodes à choisir en fonction de la difficulté intrinsèque des problèmes [1].

Les problèmes d'optimisation peuvent être divisés en deux classes, lorsqu'une solution est associée à un seul objectif, on parle de problèmes mono-objectifs, et lorsqu'elle est associée à plusieurs objectifs (souvent contradictoire, par exemple la qualité et le coût de fabrication), on parle de problèmes multi-objectifs (ou multicritères).

Les méthodes de résolution des problèmes d'optimisation peuvent être classées aussi en méthodes approchées et méthodes exactes. Les algorithmes exacts consistent en général à explorer l'espace de recherche dans sa totalité d'une façon intelligente. Ces méthodes garantissent l'optimalité, mais elles ne sont efficaces que pour les petites instances des problèmes d'optimisation. Leurs grand inconvénient est l'explosion combinatoire, ce qui rend leurs utilisation pratiquement difficile pour les grandes instances. Bien que les méthodes approchées ne garantissent pas l'optimalité d'une solution, elles sont indispensables pour les problèmes NP-difficile de grandes tailles, car généralement, elles fournissent une bonne solution dans un temps polynomial.

1.2 Notions de base

1.2.1 Diversification

La diversification (ou exploration, synonyme utilisé presque indifféremment dans la littérature) désigne les processus visant à explorer différentes zones dans l'espace de recherche du problème à optimiser [25].

1.2.2 Intensification et apprentissage

Intensification (ou exploitation) vise à parcourir une zone de l'espace de recherche pour trouver la meilleure solution. La mémoire est le support de l'apprentissage, qui permet à l'algorithme de ne tenir compte que des zones ou l'optimum global est susceptible de se trouver, évitant ainsi les optima locaux [25].

1.2.3 Voisinage

Le voisinage $v(s)$ d'une solution s est un sous ensemble de S, dont les membres sont des solutions proches (voisines) de la solution s. En effet, on dit qu'une solution « s' » est une voisine de s, si elle peut être obtenue en modifiant légèrement la solution s [2].

Etant donné, une fonction de distance dans l'espace S, $dist: S \times S \rightarrow R$ et le voisinage $v(s)$on peut définir le voisinage comme :

$$v(s) = s' \in S:\ dist(s, s') \tag{1.1}$$

1.2.4 Optimum local et Optimum global

1.2.4.1 Optimum local

Une solution $s \in S$ est un optimum local si et seulement si il n'existe pas de solution $s_0 \in v(s)$, dont l'évaluation est de meilleure qualité que s, soit :

$f(s) \leq f(s_0) \forall s_0 \in v(s)$ Dans le cas d'un problème de minimisation. (1.2)

$f(s) \geq f(s_0) \forall s_0 \in v(s)$ Dans le cas d'un problème de maximisation. (1.3)

Avec $v(s)$ l'ensemble des solutions voisines de s [1].

1.2.4.2 Optimum global

Une solution est un optimum global à un problème d'optimisation s'il n'existe pas d'autres solutions de meilleure qualité. La solution $s^* \in S$ est un optimum global si et seulement si [1] :

$f(s^*) \leq f(s) \forall s \in S$ Dans le cas de problème de minimisation (1.4)

$f(s^*) \geq f(s) \forall s \in S$ Dans le cas de problème de maximisation (1.5)

Donc, l'optimum global est la solution s^* qui vérifie la propriété précédente pour toutes les structures de voisinage du problème. La figure 1.1 schématise la courbe d'une fonction d'évaluation en faisant apparaître l'optimum global et local. [3]

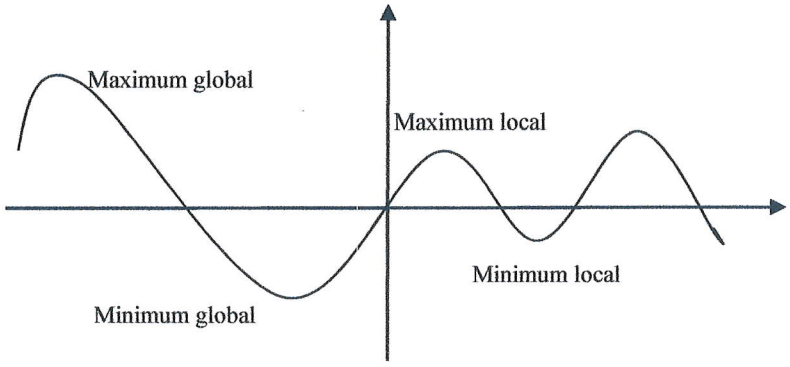


Figure 1.1 Optima locaux et optima globaux d'une fonction à une variable.

1.3 La complexité

1.3.1 Complexité d'algorithme

La complexité algorithmique est un concept fondamental qui permet de mesurer les performances d'un algorithme et de le comparer avec d'autres algorithmes réalisant les mêmes fonctionnalités [4].

En théorie algorithmique, un problème est une question générale à laquelle nous souhaitons trouver une réponse. Cette question a généralement des paramètres ou des variables dont les valeurs n'ont pas encore été déterminées. Un problème est posé en donnant une liste de ces paramètres ainsi que les propriétés auxquelles la réponse doit se conformer. Une instance d'un problème est obtenue en donnant des valeurs explicites pour chacun des paramètres du problème instancié. Un algorithme est une séquence d'opérations élémentaires (affectation de variable, tests, etc.) qui, quand on lui donne une instance d'un problème en entrée, il donne la solution de ce problème en sortie après l'exécution de l'opération finale [25].

Plusieurs types de complexité peuvent être distingués, on cite la complexité constante $o(1)$ qui est indépendante de la taille du problème, la complexité logarithmique $o(log(n))$, la complexité linéaire $o(n)$, la complexité quadratique $o(n^2)$, la complexité cubique $o(n^3)$, la complexité polynomiale $o(n^p)$, la complexité exponentielle $o(exp^n)$, et enfin la complexité factorielle $o(n!)$.

1.3.2 Complexité de problème

Afin de mesurer la difficulté d'un problème donné, la comparer avec celles d'autres problèmes pour pouvoir dire qu'un tel problème est plus facile à résoudre que l'autre, nous pouvons calculer la complexité algorithmique de chacun d'entre eux. La complexité d'un problème donné est discutée sur deux côtés : coté temporel (complexité temporelle) et coté spatial (complexité spatiale). La complexité temporelle consiste à évaluer le temps de calcul nécessaire pour résoudre un problème donné. Tandis que la complexité spatiale permet d'estimer les besoins en mémoire (l'espace mémoire requis) pour la résolution d'un problème donné [2].

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

- **Classe P** dans cette classe on trouve l'ensemble des problèmes pouvant être résolus, en temps polynomial, par une machine de Turing. La résolution de n'importe quelle instance des problèmes de cette classe se fait en un temps et espace polynomiale d'où l'appellation de classe de problèmes polynomiaux ou la Classe P.
- **Classe P-Complet** dans cette classe, on trouve l'ensemble des problèmes polynomiaux et l'ensemble des problèmes non polynomiaux où tout problème polynomial peut se réduire en un temps poly-logarithmique avec ou sans l'utilisation d'une machine de calcul déterministe.
- **Classe NP** Cette classe englobe l'ensemble des problèmes qui peuvent être résolus en un temps polynomial sur une machine de Turing
- **Classe NP-DUR** Cette classe est dite aussi NP-Difficile. Elle regroupe l'ensemble des problèmes dont la complexité est exponentielle ou doublement exponentielle. Cette classe de problème se distingue par la difficulté de résolution optimale ou exacte des grandes instances, en un temps polynomial. A cet effet, on se contente de trouver des solutions dites de bonne qualité en un temps polynomial. Parmi les problèmes de cette classe, on cite les problèmes de partitionnement d'arbres, des connexités dans un graphe de routage, Le problème de tournées de véhicules.

1.4 Méthodes de résolution des problèmes d'optimisation

Les problèmes d'optimisation combinatoires appartiennent à la famille des problèmes NP-difficiles [7], c'est-à-dire qu'on ne connaît aucun algorithme de temps polynomial pour résoudre le problème. Les petites instances de ces problèmes peuvent être résolues à

l'optimalité avec des méthodes exactes, tandis que les grandes instances ne peuvent être résolues en temps polynomial qu'avec des méthodes approchées.

1.4.1 Méthodes exactes

L'intérêt des méthodes exactes réside dans le fait qu'elles assurent l'obtention de la solution optimale du problème traité. En fait, elles permettent de parcourir la totalité de l'ensemble de l'espace de recherche de manière à assurer l'obtention de toutes les solutions ayant le potentiel d'être meilleures que la solution optimale trouvée au cours de la recherche. En effet, le temps de recherche et/ou l'espace mémoire nécessaire pour l'obtention de la solution optimale par une méthode exacte sont souvent trop grands, notamment avec des problèmes de grandes tailles. De ce fait, la complexité de ce type d'algorithme croit exponentiellement avec la taille de l'instance à traiter, elle devient très importante face à des problèmes comprenant plusieurs variables, fonctions « objectif » et/ou critères [2].

Il existe de nombreuses algorithmes exacts , l'algorithme A*, les algorithmes de séparation et évaluation (Branch and Bound , Branch and Cut, Branch and Price et Branch and Cut and Price) [2].

1.4.2 Heuristiques

Les heuristiques sont des méthodes approchées, souvent basées sur le bon sens ou sur des observations empiriques, qui permettent d'obtenir des solutions réalisables dans un temps de calcul raisonnable [6]. Une heuristique est spécifique à un problème particulier et ne peut pas être généralisée. Elle ne garantit pas l'optimalité des solutions obtenues.

Ces méthodes permettent de gérer des problèmes des grandes tailles avec des temps de résolution et des résultats acceptables et admissibles [5].

1.4.3 Métaheuristiques

Les métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonne qualité en temps raisonnable à des problèmes combinatoires réputés difficiles pour lesquels on ne connaît pas de méthode classique plus efficace. On appelle Métaheuristiques des méthodes conçues pour échapper aux minima locaux. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs qui progressent vers un optimum global [1].

Une métaheuristique est une heuristique généralisée qui peut être appliquée à plusieurs problèmes d'optimisation.

Classification des métaheuristiques

Plusieurs classifications des métaheuristiques ont été proposées, parmi ces classifications, nous pouvons citer :

Première classification : métaheuristiques à base de population ou à trajectoire

Les métaheuristiques peuvent être classées en fonction de solutions utilisées en même temps. Les méthodes à trajectoire, sont des algorithmes basées sur une solution unique à n'importe quelle étape dans l'algorithme comme les algorithmes de recherche locales tels que TS (Tabu Search) [7], ILS (Iterated Local Search) [8], VNS (Variable Neighborhood Search)[9], GRASP (Greedy Randomized Adaptive Search Procedure) [10], [11].

Les algorithmes à base de population effectuent la recherche avec plusieurs points de départ (solutions initiales) dans un style parallèle comme par exemple : ACO (Ant Colony Optimization algorithm) [12], PSO (Particle Swarm Optimization) [13], les algorithmes évolutionnistes (les algorithmes génétiques [14], la programmation génétique [Banzhaf 1998], les algorithmes mimétiques [15], les algorithmes d'évolution différentielle [16], la recherche par dispersion [17]…etc.).

Deuxième classification : métaheuristiques inspirées ou non inspirées d'un phénomène naturel

Les métaheuristiques inspirées d'un phénomène naturel peuvent se baser selon la source d'inspiration sur :

- L'intelligence en essaims comme : PSO, ACO, ABC (Artificiel Bee Colony) [18], FA (Firefly Algorithm) [19], CS (Cuckoo Search) [20], BA (Bat Algorithm) [19]…etc).
- Les systèmes biologiques comme : AG, ACO, CSA (Clonal Selection Algorithm) [21],…etc.
- La physique ou la chimie comme : Le recuit simulé (SA), HS (Harmony Search) [22], EO (Extremal Optimization) [23]…etc.

Tandis que la méthode de descente (HC) ou la recherche Tabou (TS), vont dans la seconde classe.

Troisième classification : métaheuristiques avec fonction objectif statique ou dynamique

Les métaheuristiques peuvent également être classées en fonction de la manière d'utilisation de la fonction objectif donnée dans la représentation du problème telle qu'elle,

d'autre, comme GLS (Guided Local Search) [24], la modifient lors de la recherche. L'idée dernière cette approche set de s'échapper des minimas locaux en modifiant le paysage de la recherche. En conséquence, lors de la recherche, la fonction objectif est modifiée en essayant d'intégrer les informations recueillies au cours du processus de recherche.

Quatrième classification : Les métaheuristiques avec une ou plusieurs structures de voisinage

La plupart des métaheuristiques travaillent sur une structure de voisinage unique. D'autres métaheuristiques, comme la recherche à voisinage variables (VNS), utilisent un ensemble de structures de voisinage qui donne la possibilité de diversifier la recherche et d'explorer mieux l'espace de recherche. [25].

Cinquième classification : Les métaheuristiques avec ou sans mémoire

Une caractéristique très importante pour classer les métaheuristiques est l'utililisation de l'historique de recherche. Les algorithmes qui n'utilisant pas la mémoire (historique) effectuent un processus de Markov, car l'information qu'ils utilisent pour déterminer la prochaine action est l'état actuel du processus de recherche. Il existe plusieurs façons pour utiliser la mémoire. Habituellement, nous distinguons entre l'utilisation de la mémoire à long terme et court terme. La première garde généralement la trace de mouvements effectués récemment, des solutions visitées ou, en général, les décisions prises. La seconde est généralement une accumulation de paramètres synthétique sur la recherche [25].

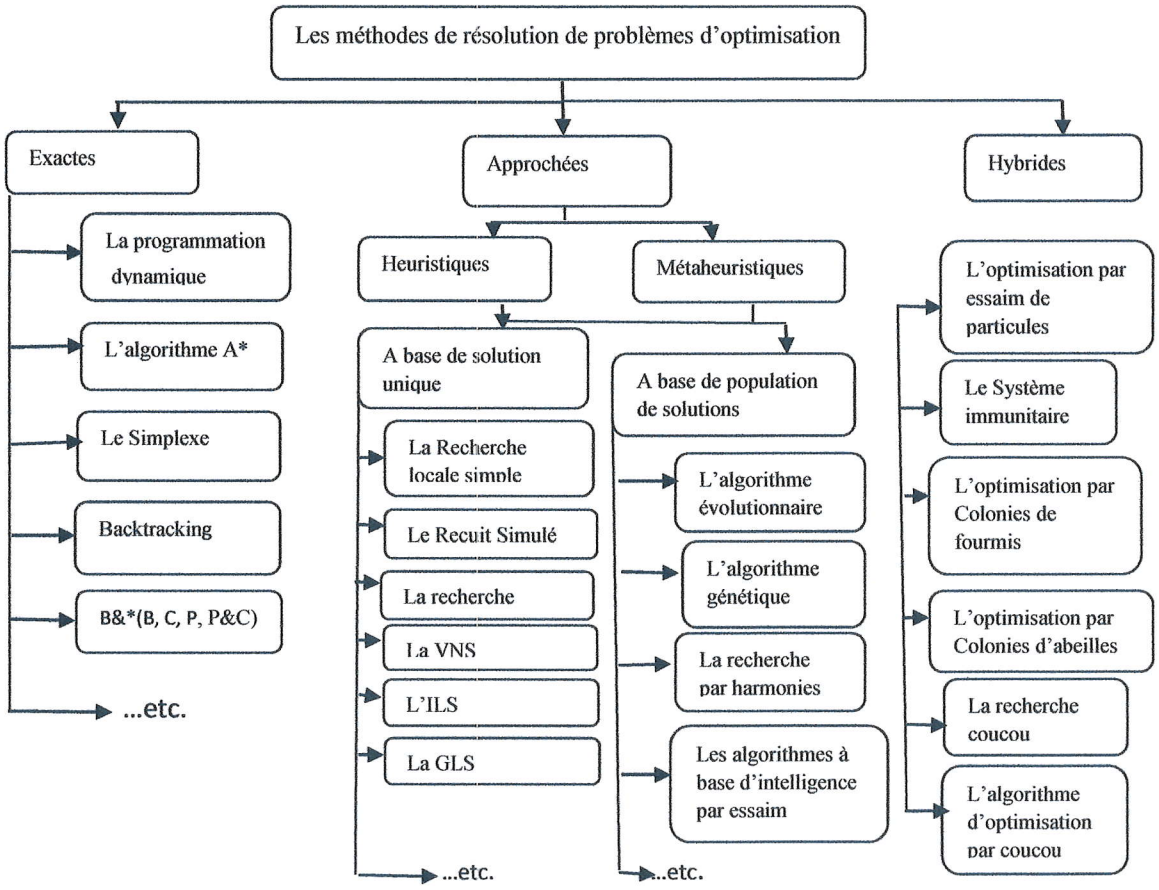


Figure 1.2 Classification de méthodes de résolution de problèmes d'optimisation.

1.5 Conclusion

La majorité des problèmes d'optimisation combinatoire appartiennent à la classe NP-DURS. Leur résolution ne peut se faire que de manière approchée, car à ce jour, il n'existe aucun algorithme permettant leur résolution en un temps polynomial.

Dans ce chapitre nous avons parlé sur les problèmes d'optimisation combinatoire. On a introduit quelques notions de bases liées au travail. Nous avons énuméré les différentes méthodes de résolution des problèmes d'optimisation à savoir les méthodes exactes, les heuristiques et les métaheuristiques. Ainsi, nous avons donné différentes classifications des métaheuristiques selon plusieurs critères. Dans le chapitre suivant, nous présentons un des problèmes d'optimisation les plus célèbres qui est le problème de tournées de véhicules.

Chapitre 2

Problème de tournées de véhicules

2.1 Introduction

Le Problème de Tournées de Véhicules (VRP, Véhicule Routing Problem) est l'un des problèmes les plus connus de la recherche opérationnelle. Suite à une collaboration très étroite entre les spécialités de la programmation mathématique et de l'optimisation combinatoire, d'une part, et les gestionnaires de transports, d'autre part, de nombreuses implantations de systèmes informatiques d'optimisation de tournées de véhicules ont vu le jour.

Parmi toutes les applications pratiques de ce problème, nous pouvons citer, entre autres, la distribution de journaux, le ramassage scolaire, la collecte d'ordures, la fourniture de combustible, la distribution de produits aux hypermarchés et magasins, la distribution de courrier, la gestion préventive d'inspection des routes,...

Historiquement, le VRP est une version étendue du problème du Voyageur de Commerce PVC, qui consiste à visiter l'ensemble des clients avec un seul véhicule. Le problème de tournées de véhicules n'est qu'une extension classique M-PVC du PVC.

2.2 Le Problème de tournées de véhicules

Le VRP (Vehicle Routing Problem) a été proposé pour la toute première fois par Lord Hamilton en 1859 [26], puis réintroduit par Dantzig and Ramser en 1959 [27]. Depuis, le VRP a fait l'objet d'études intensives.

La version basique du VRP s'énonce comme suit : une flotte de véhicules, basée dans un ou plusieurs dépôt(s), doit assurer des tournées entre plusieurs clients (ou villes) ayant demandes une certaine marchandise ou service. L'ensemble des clients visités par un véhicule désigne la tournée de celui-ci et chaque tournée commence et se termine au dépôt. Chaque client doit être desservi une et une seule fois et par un et un seul véhicule. L'objectif du VRP est de minimiser la somme des distances parcourues ou le temps total de parcours des tournées des véhicules tout en satisfaisant la demande des clients.

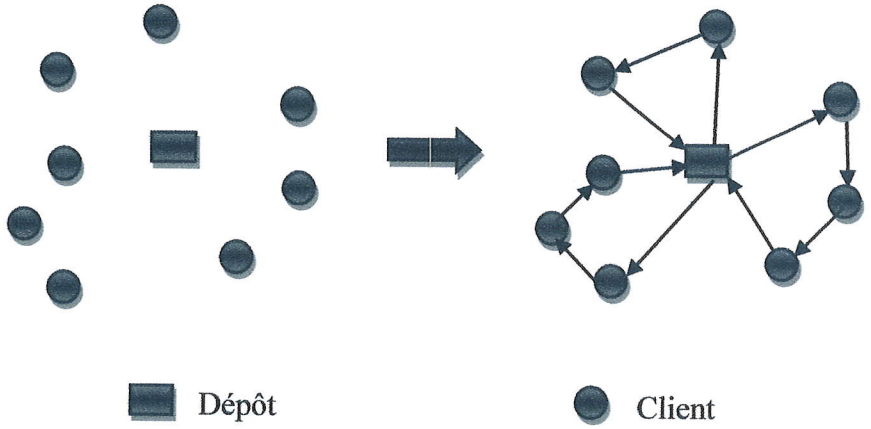


Figure.2.1 Exemple d'un problème de type VRP

2.3 Variantes de VRP

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.

2.3.1 CVRP (Capacitated Vehicle Routing Problem)

Le problème CVRP (Capacitated Vehicle Routing Problem) a d'abord été formulé par [28]. 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 [29]. Dans ce qui suit nous présentons la modélisation mathématique de CVRP proposé par Rego et Roucairol dans [30]

Soit $G = (N, A)$ un graphe où $N = 1, \dots, n$ est un ensemble de sommets avec le sommet 1 fixé comme dépôt et $A = \{(i,j) \mid i,j \epsilon N et i \neq j\}$ 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 (véhicule 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 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 [30] adopte les notations suivantes :

Les constantes de données

n = nombre de sommets,

m = nombre de véhicules,

D = capacité d'un véhicule,

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_i^j = cout ou distance du voyage du sommet$i$ au sommet j,

Les variables de décision

$$x_{ij}^k = \begin{cases} 1 & si\ le\ véhicule\ k\ voyage\ du\ sommet\ i\ au\ sommet\ j \\ 0 & sinon \end{cases} \quad (2.1)$$

Avec $X^k = (t_{ij}^k)$

La fonction à optimiser :

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \sum_{k=1}^{n} x_{ij}^k \quad (2.2)$$

Sous les contraintes :

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{ij}^k = 1, j = 2, \ldots, n \quad (2.3)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x_{ij}^k = 1, i = 2, \ldots, n \quad (2.4)$$

$$\sum_{i=1}^{n} x_{ip}^k - \sum_{j=1}^{n} x_{pj}^k = 0, k = 1, \ldots, m; p = 1, \ldots, n \quad (2.5)$$

$$\sum_{i=1}^{n} d_i(\sum_{j=1}^{n} x_{ij}^k) \leq D, k = 1, \ldots, m \quad (2.6)$$

$$\sum_{i=1}^{n} t_i^k \sum_{j=1}^{n} x_{ij}^k + \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^k x_{ij}^k \leq T_k, k = 1, \ldots, m \quad (2.7)$$

$$\sum_{j=2}^{n} x_{1j}^k \leq 1, k = 1 \ldots m \quad (2.8)$$

$$\sum_{i=2}^{n} x_{i1}^k \leq 1, k = 1, \ldots, m \quad (2.9)$$

$$X^k \in S \quad (2.10)$$

Où S donné par l'expression (2.11) [31] :

$$S = ((x_{ij}^k) | \sum_{i \in Q} \sum_{j \notin Q} x_{ij}^k \geq 1, \forall\ Q \subset N, |Q| \geq 2;\ k = 1, \ldots, m) \quad (2.11)$$

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

$$S = ((x_{ij}^k) | \sum_{i \in Q} \sum_{j \in Q} x_{ij}^k \leq |Q|-1, \forall\ Q \subset N, |Q| \geq 2;\ k = 1, \ldots, m) \quad (2.12)$$

$$S = \left((x_{ij}^k) \left| \begin{matrix} y_i^k - y_j^k + (n+1) x_{ij}^k \leq n-2, (i,j=1,\ldots,n; i \neq j; k=1,\ldots,m) \\ 1 \leq y_i^k \leq n-1, (i=2,\ldots,n) \end{matrix} \right. \right) \quad (2.13)$$

La fonction objectif (2.2) consiste à minimiser le coût total de transport. Les équations (2.3) et (2.4) assurent que chaque sommet ne soit servi qu'une seule fois par un et un seul véhicule. L'équation (2.5) assure la continuité d'une tournée par un véhicule : le sommet visité doit impérativement être quitté l'équation (2.6) assure le respect de la contrainte de capacité du véhicule. L'équation (2.7) assure le respect de la contrainte de la durée totale d'une tournée. Les équations (2.8) et (2.9) 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 (2.10), (2.11), (2.12)et (2.13) 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.

2.3.2 VRPTW (Vehicle Routing Problem with Time Windows)

Le problème de tournées de véhicules avec contrainte de fenêtre 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 [32].

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 type 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 [33]. Pour étendre le modèle proposé par Fisher et Jaikumar [74] 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 de sommet i.

t_{ij} : Le temps de transport entre les nœuds i et j.

u_{ki} : L'instant de visite du sommet i par le véhicule k.

T : Une grande valeur (T>>0).

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

$$\forall i \in (1\ldots n), \forall k \in (1\ldots M), a_i \leq u_{ik} \leq b_i \quad (2.14)$$

$$\forall i \in (1\ldots n), \forall j \in (1\ldots n), \forall k \in (1\ldots M), u_{ik} + t_{ij} - T(1 - x_{ij}^k) \leq u_{ik} \quad (2.15)$$

La contrainte (2.14) 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 (2.15) vérifié la cohérence des instants de visite lorsque deux sites se suivent.

2.3.3 VRPB (Vehicle Routing Problem with Backhauls)

Le VRPB est un 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 [34]. Les clients sont numérotées de sorte que :$L = 1, \ldots, n$ et $B = n+1, \ldots, n+m$.

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 livraisons doivent être servis avant tout client de collecte qui peut être servi. Une demande non négatived_i, pour être livrée ou collectée en fonction de ce type, est associée à chaque client$i$, et le dépôt est associé à une demande fictive $d_0 = 0$. Lorsque la matrice des couts est asymétrique, le problème est appelé Asymétrique 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 [34] :

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 de livraison précédera la clientèle de collecte, le cas échéant. Les circuits contenant seulement les clients de ramassage 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.

2.3.4 VRPPD (Vehicle Routing Problem with Pickup and Delivery)

Les problèmes de collecte et livraison (PCL) sont des problèmes d'optimisation des transports par tournées dans lesquels des passagers ou des marchandises doivent être transportés d'un point à un autre [35].

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 utilisé 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 la destination de la demande de livraison, et D_i dénote le sommet qui est la destination de la demande de collecte.

Le VRPPD est une généralisation du VRP classique, qui appartient ainsi à une large famille de problèmes de livraison et collecte. On peut être distingué trois fameux type de problèmes de collecte et livraison qui ont être 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 nœud. 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 nœud peut agir comme un nœud de livraison et de collecte au même temps. Finalement le PDP à n-produits se produit quand chaque produit est associé à un seul nœud de collecte et un seul nœud 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 [25].

2.3.5 OVRP (Open Vehicle Routing Problem)

Le problème de tournées du véhicule ouvert (OVRP) est une variante du problème de routage des véhicules classiques où les véhicules ne reviennent pas au dépôt après les services des clients. L'application d'OVRP dans la vie réelle s'agit du cas où soit l'entreprise n'a pas du tout de véhicule, 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 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 [36].

Dans le problème OVRP l'objectif est de minimiser le nombre de véhicules et de minimiser la distance totale (ou 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 des servie 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 [37].

La formulation générale est décrite comme suit [38] : soit n le nombre de clients, $N = 1 \ldots n$ est l'ensemble des clients, et $N_0 = 0, 1, \ldots, n$ l'ensemble de tous les clients et le dépôt. Ce dernier est identifié par 0. Soit $K = \{1, 2, \ldots, k\}$ l'ensemble des véhicule. Soit d_i la

demande du client $i \epsilon N$, soit l_h la charge du véhicule $h \epsilon K$. En outre, $max(d_i) \leq max(l_h)$. Définissant :

$$y_{ih} \begin{cases} 1 & si\ le\ sommet\ i(client\ ou\ depot) est\ visité\ par\ le\ véhicule\ h \\ 0 & sinon \end{cases} \quad (2.16)$$

$$x_{ijh} \begin{cases} 1 & si\ le\ véhicule\ voyage\ du\ sommet\ i\ au\ sommet\ j \\ 0 & sinon \end{cases} \quad (2.17)$$

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

$$Min\ Z = \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{h=1}^{k} c_{ij}\, x_{ijh} \quad (2.18)$$

Sous :

$$\sum_{i=0}^{n} d_i\, y_{ih} \leq l_h\ ;\ \forall\ h \epsilon K \quad (2.19)$$

$$\sum_{h=1}^{k} y_{ih} = 1,\ i = 1, \ldots, n \quad (2.20)$$

$$\sum_{i=0}^{n} x_{ijh} = y_{ih},\ j = 1, \ldots, n\ ;\ \forall\ h\ \epsilon K \quad (2.21)$$

$$\sum_{j=1}^{n} x_{ijh} = y_{ih},\ i = 1, \ldots, n\ ;\ \forall\ h\ \epsilon K \quad (2.22)$$

$$x_{ijh} = 0\ or\ 1\ ;\ i,j = 1, \ldots, n\ ;\ \forall\ h \epsilon K \quad (2.23)$$

$$y_{ih} = 0\ or\ 1\ ;\ i,j = 1, \ldots, n\ ;\ \forall\ h \epsilon K \quad (2.24)$$

Dans le modèle, la formule (2.18) est la fonction objectif qui minimise la distance totale traverse. La contrainte (2.19) porte sur la capacité des véhicules.la contrainte (2.20) indique que chaque nœud de demande doit être servi ; (2.21), (2.22) assurent que chaque nœud de demande est servi par le même véhicule.

2.3.6 HVRP (Heterogeneous Vehicle Routing Problèm)

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 chacun débute et se termine au dépôt, pour une flotte hétérogène de véhicule qui sert 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 HVRP avec flotte illimitée.

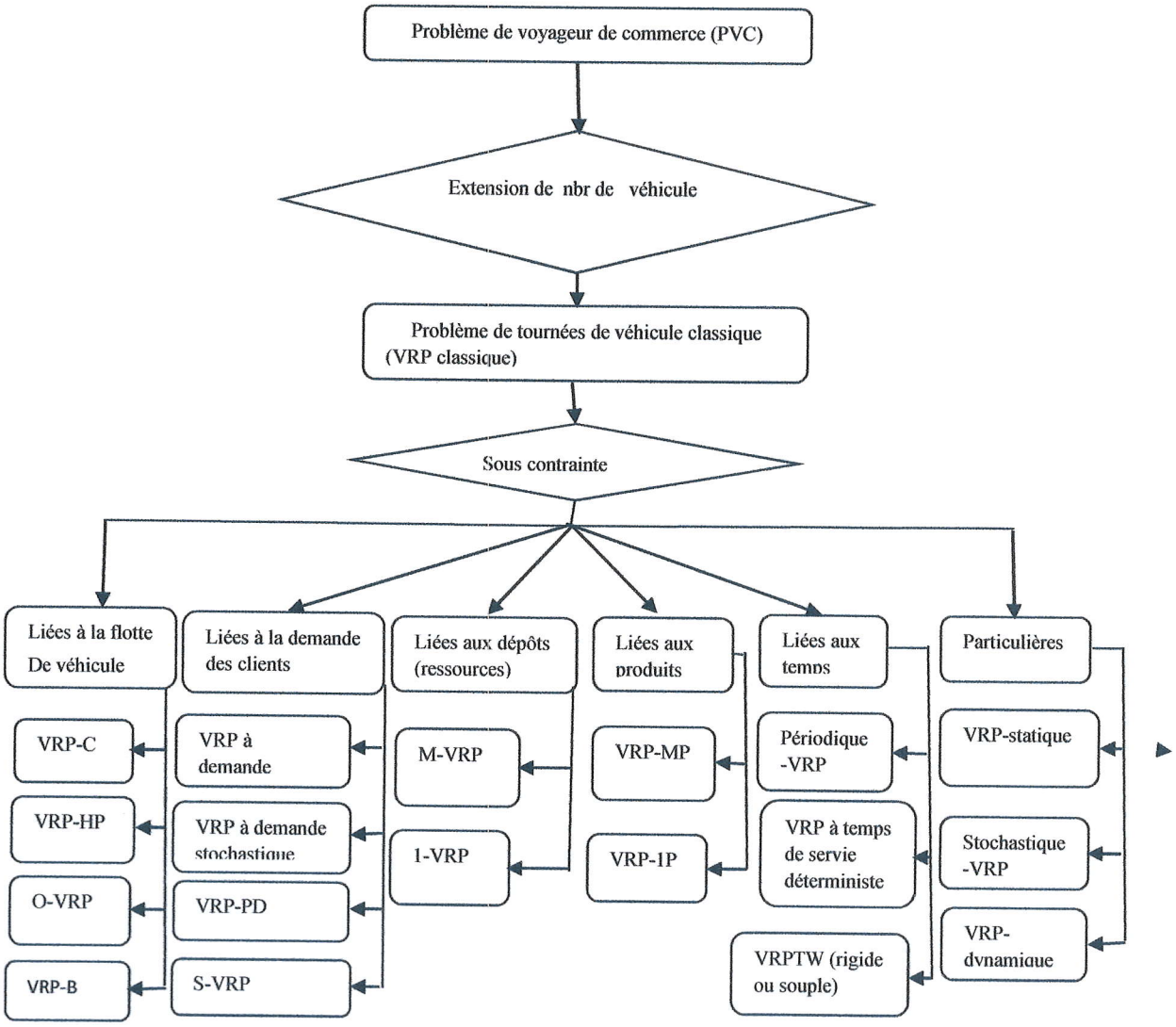


Figure.2.2 Variantes du VRP

2.4 Les méthodes de résolution de VRP et ses variantes

Le VRP appartient à la famille des problèmes NP-difficiles [75], c'est-à-dire qu'on ne connaît aucun algorithme de temps polynomial pour résoudre le problème.

- Les méthodes exactes où la solution optimale du problème est obtenue.
- Les méthodes approchées qui permettent d'obtenir de bonnes solutions sans toutefois pouvoir garantir leur optimalité.

2.4.1 Travaux sur l'état de l'art

Dans [39], quelques résultats principaux connus 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 leur livre [34].

Une autre revue sur le problème VRPPD s'est trouvée sur deux parties dans [40]. Ces dernières années ont vu une attention accrue sur le VRP intégré avec des contraintes de chargements supplémentaires, connus sous nom de 2L-CVRP ou 3L-CVRP.

Dans [41], une étude de l'art sur 2L-CVRP et 3L-CVRP est présentée. Dans [42], 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 dynamique et présentent une revue complète d'application et méthode de résolution des problèmes de tournées de véhicule dynamique. Le document [43], 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 récents 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 [44].

Compte tenu de la sensibilité environnementale des problèmes de tournées de véhicules, une revue approfondie de la littérature des problèmes de tournées de véhicules verts (Green VRP) est présenté dans [45].

2.4.2 Les méthodes exactes

En 2008, Baldacci et Mingozzi [46] 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 Depot 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 [47]. 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 [48]. 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 itératif et s'appuie sur un modèle de flux de réseau pseudo polynomiale. Dans [49], 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.

2.4.3 Les méthodes hybrides

Russell et al. [50] 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 [51] pour résoudre le CVRP. Un autre algorithme HQIEA (Hybrid Quantum Inspired Evolutionary Algorithm) est proposé dans [52], 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 [53],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 [54]. 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.

2.5 Conclusion

Dans ce chapitre nous avons présenté le problème du VRP, après une description du problème, on a présenté ses variantes avec leurs formulations mathématiques.

Nous avons cité quelques travaux dans la littérature, qui ont résolu des variantes du VRP. Le chapitre suivant entame la résolution de la variante étudié à savoir le problème de tournées de véhicule avec fenêtre de temps VRPTW.

Chapitre 3

Approche Adaptative pour la résolution du VRPTW

3.1 Introduction

Le but de ce chapitre, est d'appliquer une approche adaptative pour résoudre une variante du VRP nommée VRPTW.

La première partie comporte la présentation et la modélisation du problème de tournées de véhicules avec fenêtres de temps en spécifiant l'ensemble de ses paramètres.

Nous présentons par la suite les différentes approches d'adaptation des métaheuristiques existantes, ainsi que l'adaptation proposée dans ce mémoire en général. Nous terminons le chapitre par une présentation de notre approche adaptative conçue pour résoudre le problème de tournées de véhicules avec fenêtres de temps. Cette approche nommée EDBA (Enhanced Discret Bat Algorithm) se base sur l'algorithme de chauve-souris amélioré par le recuit simulé, en détaillant les différentes étapes qui la constituent.

3.2 Modélisation du VRPTW

Le problème de tournées de véhicules avec fenêtre de temps est parmi les variantes les plus fréquentes du VRP, en effet le VRPTW modélise de nombreux phénomènes de logistiques. En plus des définitions présentées dans le deuxième chapitre consternant le VRP et ses variantes, particulièrement celle du VRPTW, on précise que le problème de tournées de véhicules qui sera considère comme exemple d'application de notre approche est un VRPTW a contraintes de temps rigides. La flotte de véhicules est supposée homogène de taille limitée, la capacité des véhicules est identique pour l'ensemble des véhicules et elle est limitée. Les véhicules démarrent tous d'un seul dépôt et au même instant. La demande et la durée de service des clients sont connues par les distributeurs avant de quitter le dépôt.

L'objectif est de satisfaire la demande de chacun des clients à temps en parcourant une distance minimale.

Le problème du VRPTW à contraintes de temps rigides se modélise exactement comme le VRP classique ou de base présenté dans le chapitre 2 auquel nous rajoutons les contraintes relatives aux fenêtres de temps (voir section **2.3.2**).

3.3 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) [55]. 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 [56]. Comme la diversité peut être principalement 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 fort, 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 importante. 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 [57], 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 [58], 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 et peu sur la taille de la population. En outre, le papier propose un nouvel algorithme jDE auto-adaptatif en utilisant deux stratégies, sélectionnées par la même probabilité durant l'exécution. Dans [59], 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 [60] présente une revue des techniques adaptatives les plus pertinents tel que proposé dans les papiers abordant la coévolution 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 aux divers mécanismes, qui, d'une certaine manière tentent de régler automatiquement les paramètres de l'algorithme. Bien sûr cela a fait apparaitre le spectre du paramétrage inapproprié résultant d'un mauvais choix de la technique d'adaptation. Les valeurs appropriés des paramètres changeant 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 3.1 résume le réglage des paramètres par la descente.

Algorithme 3.1 Réglage des paramètres (la descente) d'une Métaheuristique M avec n paramètres

Générer une solution initiale aléatoirement $S = [Paramétre_1^*, ..., Paramétre_n^*]$

Appliquer l'algorithme M et calculer F^*

Générer K solutions voisines de solution S tel qu'une solution voisine

S_k= [$Paramétre_{1,k}$, ... ,$Paramétre_{n,k}$]

Pour chaque solution voisine K :

Appliquer l'algorithme M et calculer son F_k^*

Si $F_k^* > F^*$, $k \in$ K **alors**

$Paramétre_1^* = Paramétre_{1,k}^*$, ..., $Paramétre_n^* = Paramétre_{n,k}^*$, $F^* = F_k^*$

3.4 Résolution du VRPTW avec l'algorithme de chauves-souris

Dans ce travail, nous proposons de résoudre le problème VRPTW en utilisant un algorithme amélioré de BA. Cette amélioration est effectuée par l'application de l'algorithme de recuit simulé sur chaque solution dans chaque itération.

3.4.1 BA : Bat Algorithm (algorithme de chauves-souris)

Introduit par [61], l'optimisation par la technique des chauves-souris (*BAT*) est une approche inspirée du comportement de chasse des chauves-souris.

Au cours de leur vol et, afin d'éviter les obstacles et cibler leur proies, chaque individu (bat) émet un bisonar [62] à travers son environnement, le retour de l'écho permet d'identifier les divers objets de son entourage. Les études développées dans ce domaine [63] ont montré que la résonance de l'onde émise varie d'une valeur élevée au cours d'un vol de prospection à une valeur assez faible avec une augmentation de fréquence quand la chauve-souris détecte et oblique en direction d'une proie. Il est supposé par intuition que ces animaux sont en mesure de différencier leurs proies des autres obstacles proches, notamment les chauves-souris avoisinantes.

L'algorithme BAT peut être implémenté pour la résolution des problèmes d'optimisation continue où les solutions possibles peuvent être représentées par les positions géographiques des bats.

Le principe de l'algorithme est décrit comme suit :

Dans un espace $\mathbb{R}^n$, à l'instant t, chaque bat i a une position x_t^i se déplaçant à une vitesse $v_t^i \in \mathbb{R}^n$ en émettant une pulsation de fréquence constante f_{min}.

A la perception d'une proie, les paramètres de position et vitesse seront ajustés selon les relations suivantes :

$$f_i = f_{min} + \beta(f_{max} - f_{min}) \quad (3.1)$$

$$v_i^{t+1} = v_i^t + f_i(x_i^t - x_g^t) \quad (3.2)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3.3)$$

Où β, un vecteur aléatoire $\in [0,1]$ et x_g^t, la position de la meilleure chauve-souris du groupe.

L'algorithme illustrant le mouvement des chauves-souris est présenté dans l'algorithme (Algorithme 3.2).

Algorithme 3.2 : Algorithme BAT

1. Fonction objectif $f(x), x = (x_1, \dots, x_d)$
2. Initialiser la population de chauves-souris $x_i (i = 1, 2, \dots, n)$ et v_i
3. Définir la fréquence d'impulsion f_i de x_i
4. Initialiser le taux d'émission d'impulsions r_i et de bruit A_i
5. **tant que**$(t < MaxNombred'itérations)$
6. Générer de nouvelles solutions en ajustant la fréquence
7. mise à jour des vitesses et emplacements/solutions [les équations (1) à (3)]
8. **Si**$(rand > r_i)$
9. Sélectionnez une solution parmi les meilleures solutions
10. Générer une solution locale autour de la meilleure solution sélectionnée
11. **fin Si**
12. Générer une nouvelle solution en volant au hasard
13. **Si**$(rand < A_i \& f\ (x_i)\ < f\ (x^*\ ))$
14. Accepter les nouvelles solutions
15. Augmenter la r_i et réduire A_i
16. **fin Si**
17. Classer les chauves-souris et trouver la meilleure x^*
18. **fin tant que**
19. Processus post résultats et visualisation

3.4.2 EDBA : Algorithme amélioré de chauve-souris discret pour la résolution de VRPTW

En général, les problèmes d'optimisation peuvent être divisés en deux catégories principales, continues et discrètes [64, 65].Un problème d'optimisation avec les variables de décision réelles est connu comme un problème d'optimisation continu, alors qu'il est appelé un problème optimisation discrète si les variables de décision prennent des valeurs discrètes

(habituellement entières). En outre, dans les problèmes discrets, l'ensemble des solutions possibles est discret ou peut être réduit à une discontinue par la discrétisation de l'espace continu.

Fondamentalement, le BA a été développé pour optimiser les fonctions non linéaires continues [66, 67]. Chaque bat se déplace dans l'espace de recherche vers une position valorisée en continu, mais de nombreux problèmes sont, cependant, définis dans des espaces valorisés discrets où le domaine des variables est fini. Généralement, plusieurs algorithmes bio-inspirés à base de population ont illustré une bonne efficacité dans la résolution de problèmes continus ainsi que les problèmes discrets comme BA [66], CS (Cuckoo Search) [68], et PSO (Particle Swarm Optimisation) [70].

Dans les versions originales de ces algorithmes, il est impossible de les exploiter directement pour résoudre problèmes discrets, nombreux chercheurs ont modifié ces derniers pour traiter des problèmes discrets et nous pouvons en citer quelques-uns : Binary Bat Algorithm (BBA) [71], Discrete Cuckoo Search (DCS) [69], and Discrete Binary Particle Swarm Optimisation (DBPSO) [72].

Nous proposons de résoudre le VRPTW en utilisant un algorithme amélioré de BA (Bat Algorithm) appelé EDBA. Cette amélioration est effectuée par l'application de l'algorithme de recuit simulé à chaque solution dans chaque itération. Cette approche enregistre le concept original et les mêmes étapes de base proposées est présentées dans BA, mais modifie certaines équations de passage en espace discrets. Pour les sous-sections suivantes nous décrivons brièvement les différentes étapes de l'extension de la version de BA pour résoudre le problème VRPTW.

3.4.2.1 Représentation de position et vitesse

Dans l'algorithme basique BA, la position x_i de chaque chauve-souris définit une solution potentielle au problème, pour trouver la meilleure solution, chaque chauve-souris ajuste sa vitesse en sélectionnant au hasard une fréquence f_i . D'autre part, chaque chauve-souris utilise le taux d'émission d'impulsions r_i et loudness A_i pour contrôler l'intensité de recherche locale qui est traitée pour produire un nouvel individu autour d'une des meilleures solutions globales.

3.4.2.2 Équation de mise à jour de position

Dans l'algorithme BA continu, le déplacement des chauves-souris est calculé par les trois équations (3.1), (3.2), (3.3). En utilisant les équations (3.1) et (3.2), chaque chauve-souris met à jour sa vitesse. La valeur obtenue est utilisée dans l'équation (3.3) pour calculer la

position suivante de chaque chauve-souris. Toutefois, afin d'appliquer BA pour résoudre le problème de VRPTW, ces équations ne peuvent pas être utilisées directement et doivent être adaptées au problème.

Comme dans l'algorithme BA classique, chaque chauve-souris sélectionne une fréquence f_i dans l'intervalle de fréquence $[f_{min}, f_{max}]$, où f_{min} et f_{max} sont deux nombres entiers appartenant à l'intervalle $[1,n]$, n représente le nombre de clients. La fréquence f_i indique le nombre de clients enregistré à partir de la solution actuelle x_i^t lorsque cette dernière est croisée avec la meilleure solution globale actuelle x_*^t. La vitesse consiste à effectuer un ensemble de permutations qui permettent le croisement de deux solutions x_i^t et x_*^t

3.4.2.3 La recherche de voisinage

La recherche de voisinage est utilisée pour améliorer une solution itérativement, dans le VRPTW, cela consiste à trouver une meilleure solution dans le voisinage des solutions en effectuant des changements minimaux.

Une des procédures les plus connues dans la recherche locale est la méthode 2-opt où deux arêtes sont échangées de manière itérative jusqu'à ce qu'aucune autre amélioration n'est possible.

3.4.3 Algorithme EDBA

L'algorithme EDBA développé dans le cadre de ce problème noté « algorithme 3.6 » suit les étapes suivantes :

1- Fixation des paramètres de l'algorithme en utilisant la technique d'adaptation proposée.
2- Initialisation des chauves-souris (population initiale).
3- mise à jour des chauves-souris (permutation basé sur la méthode 2-échange).
4- générer solution locale en utilisant la méthode 2-opt.
5- Intensification par le recuit simulé (RS).

Algorithme 3.3 : Algorithme EDBA

---

1. Réglage des paramètres : $nombre\ d'itérations, taille\ de\ la\ population,\ \gamma,\ \alpha.$
2. Création de la population initiale de taille $(tailledelapopulation)$ des chauves-souris satisfaisant les contraintes de problème.
3. Définir la fréquence d'impulsion f_{min} et f_{max} dans la gamme $[1,n]$.
4. Initialiser les paramètres $r_i \in [0.0\ ,1.0]$ et $A_i \in [0.0\ ,1.0]$.
5. calcul de fitness de chaque chauve-souris.

6. **tant que**($t < nombre\ d'itérations$).
7. S^* = meilleure chauve-souris (meilleure solution).
8. **pour** chaque chauve-souris (solution) s^ide population **faire :**
9. Mise à jour de chauve-souris (croisement 2-échange S^* avec s^i basée surf_i) tout en satisfaisant les contraintes du problème
10. **Si**($rand > r_i$)
11. Sélectionner une solution S^{meil} parmi les meilleures solutions
12. Générer une solution S^L voisine de S^{meil} en appliquant la méthode 2-opt et en satisfaisant les contraintes de problème.
13. **Si**($rand < A_i \& f\ (S_i)\ < f\ (S^*\ )$) :
14. Accepter les nouvelles solutions.
15. Augmenter r_i et réduire A_i (mise à jour basée surγ,α).
16. **fin Si.**
17. **fin pour.**
18. Classer les chauves-souris et trouver la meilleure solution courante S^*
19. Appliquer le recuit simulé sur toutes les solutions.
20. **fin tant que**

3.4.3.1 Encodage de la solution

Le codage associe à une solution une structure de données. Elle se place généralement après une phase de modélisation du problème traité. 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 N, chaque élément de ce vecteur est un vecteur de taille M, tel que N représente le nombre total des tournées et M représente le nombre total des clients dans une tournée. 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 de départ et en terminant à ce dépôt.

Exemple

Supposant qu'on ait 10 clients (codés de 1 à 10). 0 représente le dépôt. Une solution faisable est représentée dans le tableau « Table 3.1». Le nombre de véhicules représente le nombre de tournée et égal à la taille de vecteur.

Cette solution utilise quatre véhicules.

v1	v2	v3	v4
0	0	0	0
1	8	2	3
5	9	7	10
4	0	6	0
0		0	

Table 3.1 Exemple d'une solution faisable

3.4.3.2 Description de l'algorithme

Création de la population initiale des chauves-souris

Nous générons aléatoirement un ensemble initial de chauve-souris (solutions faisables) chaque chauve-souris code une solution possible (algorithme 3.4).

Algorithme3.4 Création de la population initiale

S = solution vide (vecteur).

Répéter

T = nouvel tournée vide (vecteur).

Insérer le dépôt dans T.

Pour chaque client resté **faire**

Si les contraintes de capacité et de temps de fenêtre sont vérifiées **Alors**

Le client sélectionné est insérée dans la tournée T.

Fin pour

Insérer la tournée T à la solution S.

Jusqu'à Tous les clients sont servis

Toutes les tournées se terminent au dépôt (0).

Calcul de fitness de chaque chauve-souris

Après la création de l'ensemble des chauves-souris, chaque chauve-souris est évaluée et se voit assignée une valeur de fitness, selon la fonction de fitness suivante :

$$fitness = \sum_{i=1}^{n} distanceparcourueparlatournéei \quad (3.7)$$

Croisement 2-échange basé sur f_i

L'opérateur de mise à jour basé sur la fréquence f_i est décrit dans l'algorithme 3.5. Il est utilisé pour générer une nouvelle solution en appliquant un ensemble de permutation entre la meilleure solution avec chaque solution de la population (en basant sur la méthode 2-échange).

Exemple

Soient S^* la meilleure solution courante et S^i la $i^{éme}$ solution de la population. S^*et S^isont deux solutions générées dans la même itération. Nous décrivons dans ce qui suit le processus de méthode 2-échange dans cet exemple nous prenons $f_i = 4$.

S^*	0	5	1	0	6	7	0	3	4	2	0	S^i	0	4	7	0	6	1	5	0	2	3	0

$f_i = 4$, S^i : *une solution* , S^* : *meilleure solution*

S^{new}	0	4	7	0	6	1

Distance (1,5) ≤ Distance(1,5) **et** les contraintes de capacité et de temps de fenêtre sont vérifiées **alors** ajouter le client 5 à S^{new}.

S^{new}	0	4	7	0	6	1	5

S^*	0	5	1	0	6	7	0	3	4	2	0	S^i	0	4	7	0	6	1	5	0	2	3	0

S^i : *une solution* , S^* : *meilleure solution*

S^{new}	0	4	7	0	6	1	5

Distance(5,3) < Distance(5,2) **et** les contraintes de capacité et de temps de fenêtre sont vérifiées **alors** ajouter le client 3 à S^{new}.

S^{new}	0	4	7	0	6	1	5	0	3

Contrainte de capacité n'est pas vérifiée alors créer une nouvelle tournée et continuer.

S^*	0	5	1	0	6	7	0	3	4	2	0	S^i	0	4	7	0	6	1	5	0	2	3	0

S^{new}	0	4	7	0	6	1	5	0	3

Distance(5,2) = Distance (5,2) **et** les contraintes de capacité et de temps de fenêtre sont vérifiées **alors** ajouté client 2 à S^{new}.

S^{new}	0	4	7	0	6	1	5	0	3	2	0

Tous les clients sont servis : fin de 2-échange.

Figure 3.1 Exemple de la méthode 2-échange pour VRPTW.

La nouvelle solution S^{new} enregistre les 4 premiers clients de S^i. Le mécanisme de cette heuristique consiste à concaténer le dernier client de S^{new}ajouté dans la nouvelle solution avec le client le plus proche de lui entre premier client dans S^*et premier client dans S^i tout en respectant les contraintes de problème.

Algorithme 3.5 : méthode 2-échange

S^* = meilleure solution.

S^i = $i^{éme}$solution de la population.

f_i = fréquence de chauve-souris ($i^{éme}$solution).

S^{new} = nouvelle solution vide.

Enregistrer les$f_i$premiers clients de S^i dans S^{new}.

Répéter

C = dernier client dans S^{new}.

C1 = premier client dans S^i.

C2 = premier client dans S^* .

Si distance(**C, C1**)>distance(**C, C2**) **et** les contraintes de capacité et de temps de fenêtre sont vérifiées **Alors**

Ajouter C2 dans S^{new}.

Sinon

Ajouter C1 dans S^{new}.

FinSi

Jusqu'à tous clients sont servis.

Générer une solution voisine (la recherche locale)

A partir d'une solution réalisable, on applique plusieurs mouvements inter et intra-tournée pour trouver le minimum local. Nous utilisons 2 types de mouvement : 2-Opt et les permutations. Nous distinguons les mouvements internes qui ne modifient qu'une tournée et les mouvements externes qui nécessitent deux tournées. Nous appliquons ces mouvements pour trouver le voisin d'une solution. Cette dernière est sélectionnée aléatoirement parmi la meilleure moitié de la population à chaque itération.

L'opérateur de génération d'une solution voisine a tendance d'augmenter la diversité de la population en prospectant d'autres régions de l'espace de recherche.

✓ 2-Opt interne

Cette méthode (figure 3.2) découpe deux arcs d'une même tournée. On parcourt dans l'ordre de la séquence les arcs pour sélectionner successivement le premier arc, puis le second arc parmi les arcs suivants du premier. Cette opération permet d'inverser le segment à l'intérieur d'une tournée.

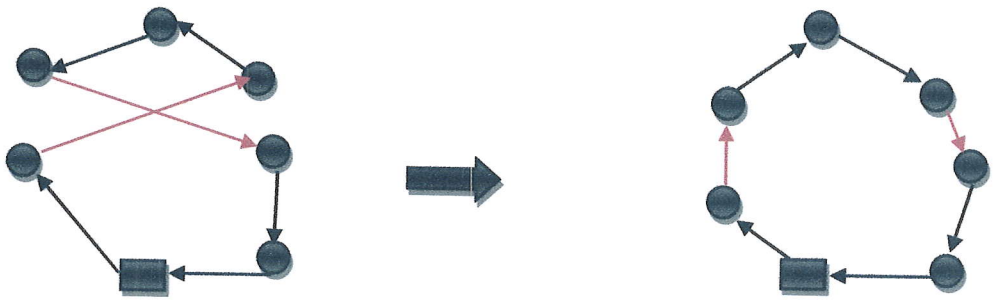


Figure 3.2 principe2-Opt interne d'une tournée.

Algorithme 3.6 Méthode de 2-Opt interne

Tournée T

Amélioration = vrai

Tant que Amélioration = vrai **faire**

Amélioration = faux

Pour tout client c_i de T **faire**

Pour tout client c_j de T, avec j différent de $i-1$ et $i+1$ **faire**

Si distance(c_i, c_{i+1}) + distance(c_j, c_{j+1}) < distance(c_i, c_j) + distance(c_{i+1}, c_{j+1}) **et** les contraintes de capacités et de temps de fenêtre sont vérifiées **alors**

Remplacer les arêtes (c_i, c_{i+1}) et (c_j, c_{j+1}) par (c_i, c_j) et (c_{i+1}, c_{j+1})

Amélioration = vrai

Fin si

Fin pour

Fin pour

Fin tant que

✓ 2-Opt externe

Cette méthode (figure 3.3) travaille sur deux tournées différentes. On sélectionne un arc dans chaque tournée. Cette opération permet d'intervertir le segment de fin entre deux tournées. Avec une configuration spéciale du 2-Opt Externe, on peut concaténer 2 trajets.

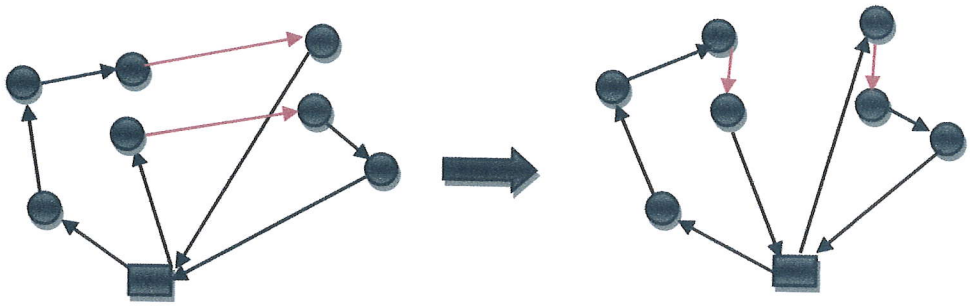


Figure 3.3 principe2-Opt externe entre deux tournées.

Algorithme 3.7 Méthode de 2-Opt externe

Tournée $T1$

Tournée $T2$

Amélioration = vrai

Tant que Amélioration = vrai **faire**

Amélioration = faux

Pour tout client c_i de tournée $T1$**faire**

Pour tout client c_j de tournée $T2$ **faire**

Si distance(c_i,c_{i+1}) + distance(c_j,c_{j+1}) >distance(c_i,c_j) + distance(c_{i+1},c_{j+1}) **et** les contraintes de capacités et de temps de fenêtre sont vérifiées **alors**

Remplacer les arêtes (c_i,c_{i+1}) et (c_j,c_{j+1}) par (c_i,c_j) et (c_{i+1},c_{j+1})

Amélioration = vrai

Fin si

Fin pour

Fin pour

Fin tant que

✓ Permutations internes

Dans cette méthode, les voisins sont générés par permutation des clients au sein d'une même tournée, ceci revient à modifier l'ordre de visite des clients par le véhicule.

La (figure 3.4) illustre le principe de cette méthode.

Figure 3.4 principe de permutation interne d'une tournée

Algorithme 3.8 : Mouvement de permutation de clients dans une même tournée

---

Choisir aléatoirement un véhicule V^id'une solution S

Choisir aléatoirement deux clients cj, ck dans V^i

Permuter cj et ck

Si vérifier Solution (S)=vrai **alors**

Retourner vrai

Sinon

Retourner faux ;

✓ Permutations externes

Cette méthode consiste à permuter deux clients choisis aléatoirement entre deux tournées différentes tout en vérifiant les contraintes de capacité et de fenêtre de temps. La (figure 3.5) illustre le mécanisme de cette permutation.

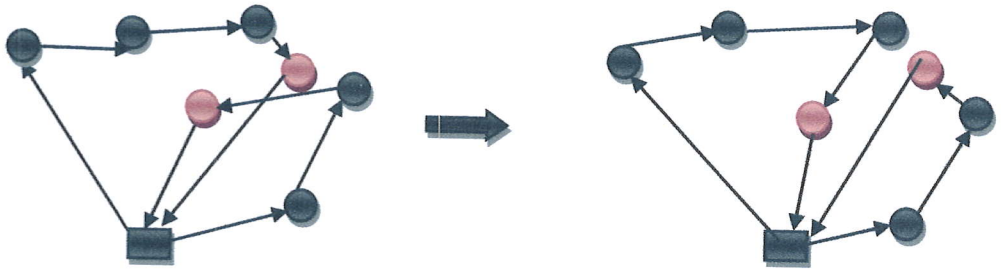


Figure 3.5 principe de permutation externe entre deux tournées.

Algorithme 3.9 Mouvement de permutation de clients entre deux tournées

Choisir aléatoirement deux véhicules V^i et V^j d'une solution S

Choisir aléatoirement deux clients, cjdans V^i et V^j

Permuter ci et cj

Si vérifier Solution (S) **alors**

Retourner vrai

Sinon

Retourner faux

Mise à jour basée sur γ,α

Les deux paramètres γ etα sont deux constantes générées par la technique d'adaptation dans la phase de fixation des paramètres. Nous appliquons les équations (3.5) et (5.6) pour augmenter r_i et réduire A_i. Cette opération est appliquée si la condition de ligne 13 de l'algorithme 3.3 est vérifiée.

Application de recuit simulé

Cette méthode de recherche a été proposée par des chercheurs d'IBM qui étudiaient les verres de spin. Ici, on utilise un processus métallurgique (le recuit) pour trouver un minimum. En effet, pour qu'un métal retrouve une structure proche du cristal parfait (l'état cristallin correspond au minimum d'énergie de la structure atomique du métal), on porte celui-ci à une température élevée, puis on le laisse refroidir lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement [73].

Nous appliquons le recuit simulé pour l'amélioration des solutions, à chaque itération pour toutes les solutions de la population, en se basant sur les différents mouvements élémentaires déclarer ci-dessus, ces mouvements élémentaires (permutation externe, permutation interne et les deux méthodes 2-opt) ont été utilisés pour améliorer le voisinage d'une solution utilisée pour la méthode recuit simulé, l'algorithme (algorithme 3.10) montre le principe de cette méthode.

Algorithme 3.10 Recuit simulé.

Initialise T (température initiale)

Initialise S (solution de départ)

Initialise ΔT (température)

Répéter

$Y =$ Voisin(S)

$\Delta C = \Delta(y) - \Delta(S)$

Si $\Delta T\ C < 0$ **Alors**

$y = S$

Sinon

$p = e^{\frac{-\Delta c}{T}}$

$r =$ valeur aléatoire dans [0,1]

Si $r < p$ **Alors**

$Y = S$

$T = \alpha\ (T)$

Jusqu'a ($T > \Delta T$)

3.5 Conclusion

Dans ce chapitre, nous avons décrit la technique d'adaptation concernant l'ajustement des paramètres d'une métaheuristique. Nous avons présenté ensuite le probléme à résoudre (VRPTW) et l'algorithme de chauve-souris (BA) qui fondamentalement développé pour optimiser les fonctions non-linéaires continues, par la suite, nous avons proposé une métaheuristique appelée (EDBA) pour résoudre le probléme de VRPTW, Cette approche enregistre le concept original et les mêmes étapes de base proposées est présentées dans BA, mais modifie certaines équations de passage en espace discrets et enrichie par une amélioration basée sur la méthode de recuit simulé.

Dans le chapitre suivant, nous présentons les résultats d'expérimentation de notre approche proposée.

Chapitre 4

Résultats et expérimentation

4.1 Introduction

Dans le chapitre précédent, nous avons développé notre approche adaptative pour la résolution du VRPTW basé sur l'algorithme de chauve-souris amélioré (EDBA).

Dans ce chapitre, nous présentons tout ce qui concerne notre expérimentation, on débute par la description des jeux de données utilisés ensuite la présentation de l'interface de l'application. Ce chapitre est clôturé par une analyse des résultats obtenus. L'analyse est basée sur la comparaison des BKS (Best Known Solutions) et les résultats obtenus par notre approche EDBA développée.

4.2 Environnement de programmation

Nous avons développé l'application en utilisant l'environnement de programmation : Netbeans IDE 8.1, sous système d'exploitation Windows 7.

4.3 Choix du langage

Le langage Java a été créé en 1991, par la firme Sun Microsystems, qui souhaitait créer un environnement indépendant du hardware et pouvant permettre de programmer des appareils.

Définition de SUN

« Java est un langage simple, orienté objet, distribué, robuste, sûr, indépendant des architectures matérielles, portable, de haute performance, multithread et Dynamique ».

Simple : inspiré du C++, Fortran, Lisp, Smalltalk. Pas de pointeur ; pas de surcharge d'opérateur, pas d'héritage multiple et présence d'un « garbage collecteur ».

Orienté objet : la programmation objet modélise des objets ayant un état (ensemble de variables) et des méthodes (fonctions) qui leur sont propres. L'unité de base en Java est la classe. Un des intérêts de Java est de disposer de nombreuses classes déjà faites.

Un objet créé à partir d'une classe est une *instance*.

Distribué : les fonctions d'accès au réseau et les protocoles Internet les plus courants sont intégrées.

Robuste : typage des données très strict, pas de pointeur.

Sûr : Java n'est pas compilé à destination d'un processeur particulier mais en « byte code » qui pourra ensuite interpréter sur une machine virtuelle Java (JVM= Java Virtual Machine),

Le « byte code » généré est vérifié par les interprétateurs Java avant l'exécution, un débordement de tableau déclenchera automatiquement une exception et l'absence d'arithmétique de pointeur évite les malversations.

Portable : les types de données sont indépendants de la plateforme (par exemple les types numériques sont définis indépendamment du type de plateforme sur laquelle de type code sera interprétée).

Haute performance : discutable car Java est un langage pseudo interprété et les techniques de « Just in time » (Jit) peuvent améliorer ces performances.

Multithread : une application peut être décomposée en unité d'exécution fonctionnant simultanément.

Dynamique : les classes Java peuvent être modifiées sans avoir à modifier le programme qui les utilise.

Politique : Java actuellement totalement contrôlé par SUN.

Définition de Netbeans:

Netbeans est un environnement de développement intégré (EDI) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, Netbeans permet également de supporter différents autres langages, comme Python, C, C++, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). Conçu en Java, Netbeans est disponible sous Windows, Linux, Solaris ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Il constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE Netbeanss'appuie sur cette plateforme.

Nous avons utilisé le langage Java et le Netbeans IDE pour coder notre algorithme.

4.4 Les jeux de données

Les instances de VRPTW construites par Solomon 1987 sont disponibles sur le site http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-with-time-windows-instances/

Ces instances se limitent à 100 clients, elles se répartissent en six (6) catégories selon le processus suivi pour la détermination des positions des clients ainsi que la demande des clients et les fenêtres de temps. Ils font ressortir plusieurs paramètres qui influent sur le comportement des algorithmes de résolution de problèmes de routage. Les coordonnées géographique des clients et du dépôt sont générées aléatoirement pour les problèmes des

catégories R1 et R2, par contre les problèmes des catégories C1 et C2 sont générées de façon à assurer le regroupement des clients en sous-ensembles ou en clusters. Les problèmes des catégories RC1 et RC2 sont structurés de sorte à garantir le positionnement aléatoire et regroupé.

Ces trois types de répartitions sont illustrés dans la figure 4.2 qui visualise les instances C101, R101 et RC101.

Les classes C1-R1-RC1 ont un horizon temporel court et des routes courtes (de5 à 10 clients). Elles nécessitent entre 9 et 19 véhicules de petite capacité. Les classes C2-R2-RC2 ont, elles, un horizon temporel long, des routes plus longues (certaines routes pouvant atteindre 30 clients) et nécessitent peu de véhicules (entre 2 et 4).

Nous avons récapitulé toutes ces propriétés dans le tableau 4.1.

La position des clients est désignée par les coordonnées (x, y), déterminées aléatoirement ou bien de telle sorte qu'elles permettent une répartition en groupe (cluster). La distance inter-client est déterminée par le calcul de la distance euclidienne entre les différents clients.

Pour tester notre algorithme nous avons considéré six (6) instances (une instance de chaque catégorie).pour les 3 problèmes (problème de 25 clients et problème de 50 clients et problème de 100 clients).

Les instances de Solomon doit être représenté dans un fichier texte et organisé comme suit (voir la figure 4.1).

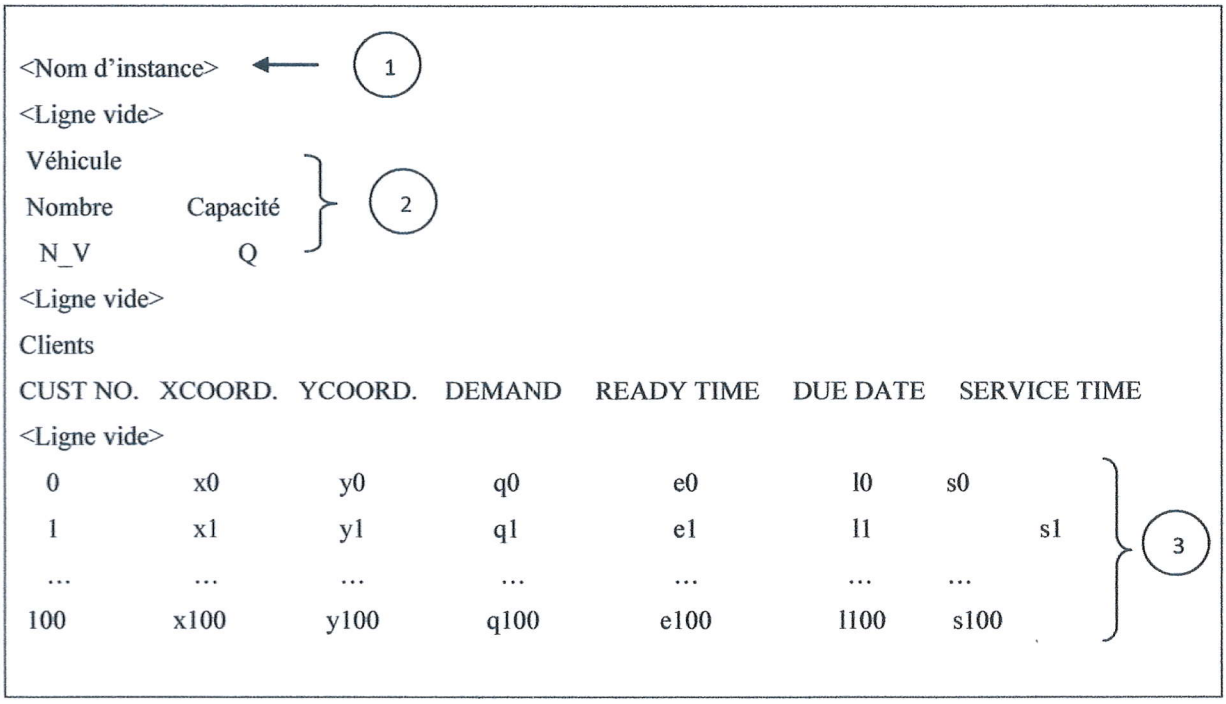
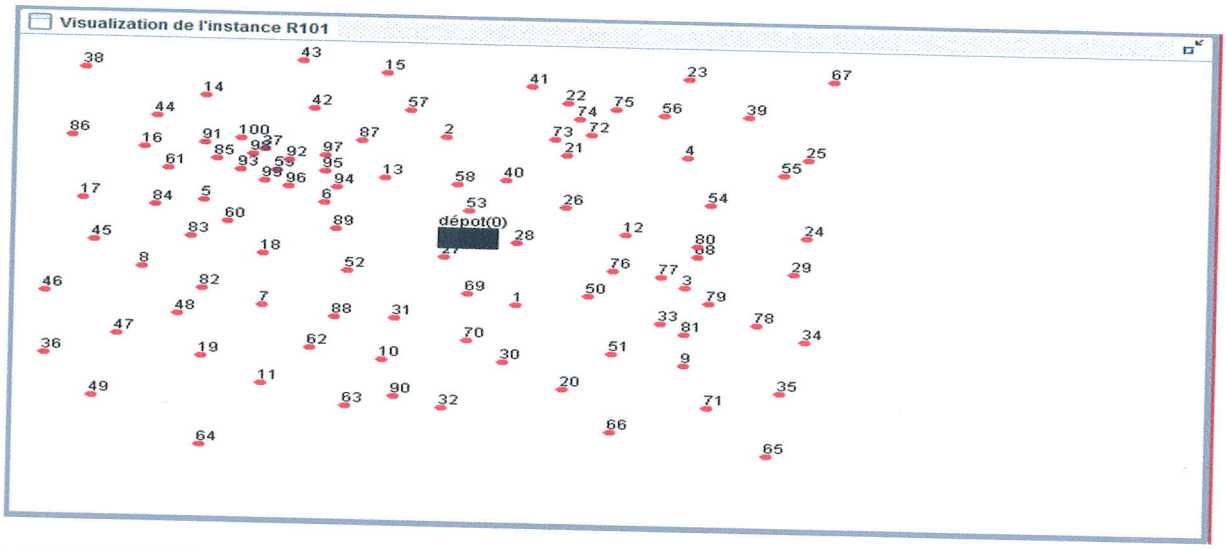


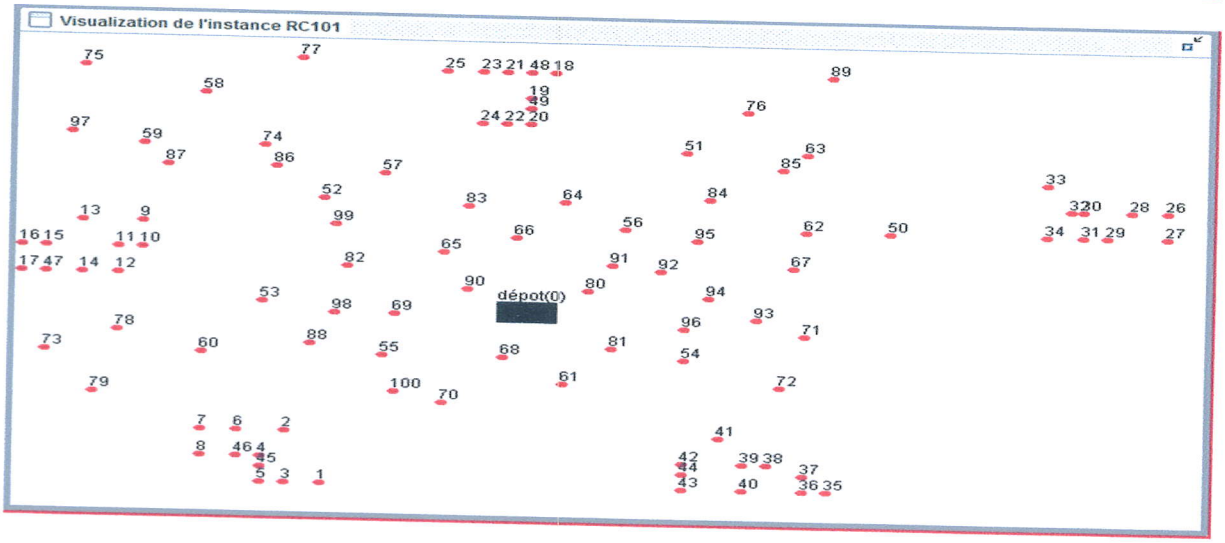
Figure 4.1 l'organisation textuelle d'une instance de Solomon.

1 : Nom de l'instance.

2 : Propriétés des véhicules {Q : capacité de véhicules, N_V : nombre de véhicules}.

3 : Propriétés des clients {Les numéros des clients (CUST NO), Les coordonnées (XCOORD, YCOORD), Les demandes de clients (DEMAND), Les fenêtres de temps (READY TIME, DUE DATE), et le temps de service des clients (SERVICE TIME)}. Le (CUST NO = 0) Représenté le dépôt.





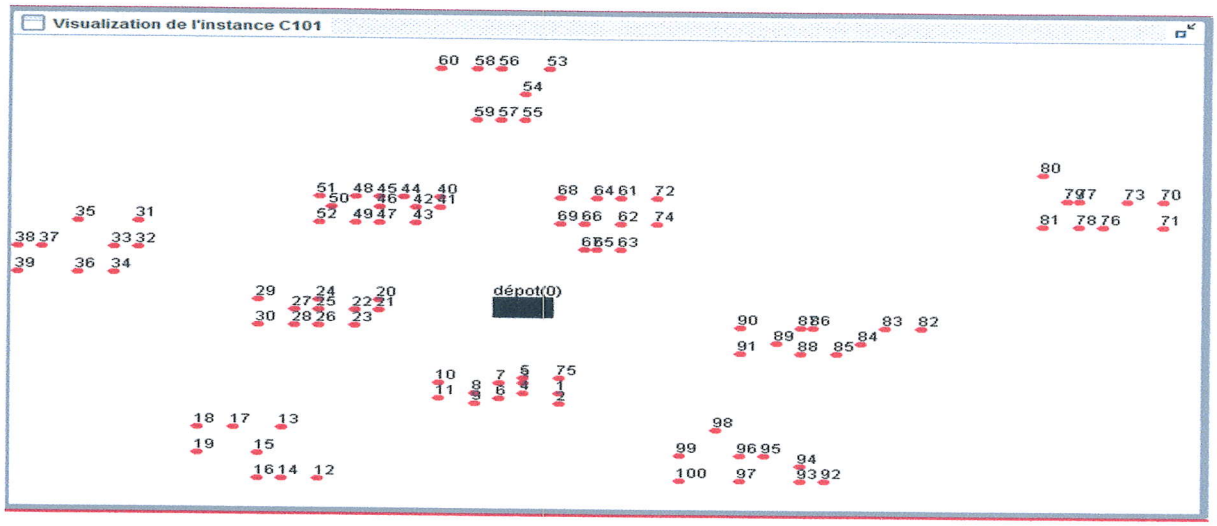


Figure 4.2 Visualisation de trois instances (C101, R101, RC101) des problèmes de Solomon

4.5 Présentation de l'interface

Au lancement de l'exécution de l'application la fenêtre (figure 4.3) s'apparue.

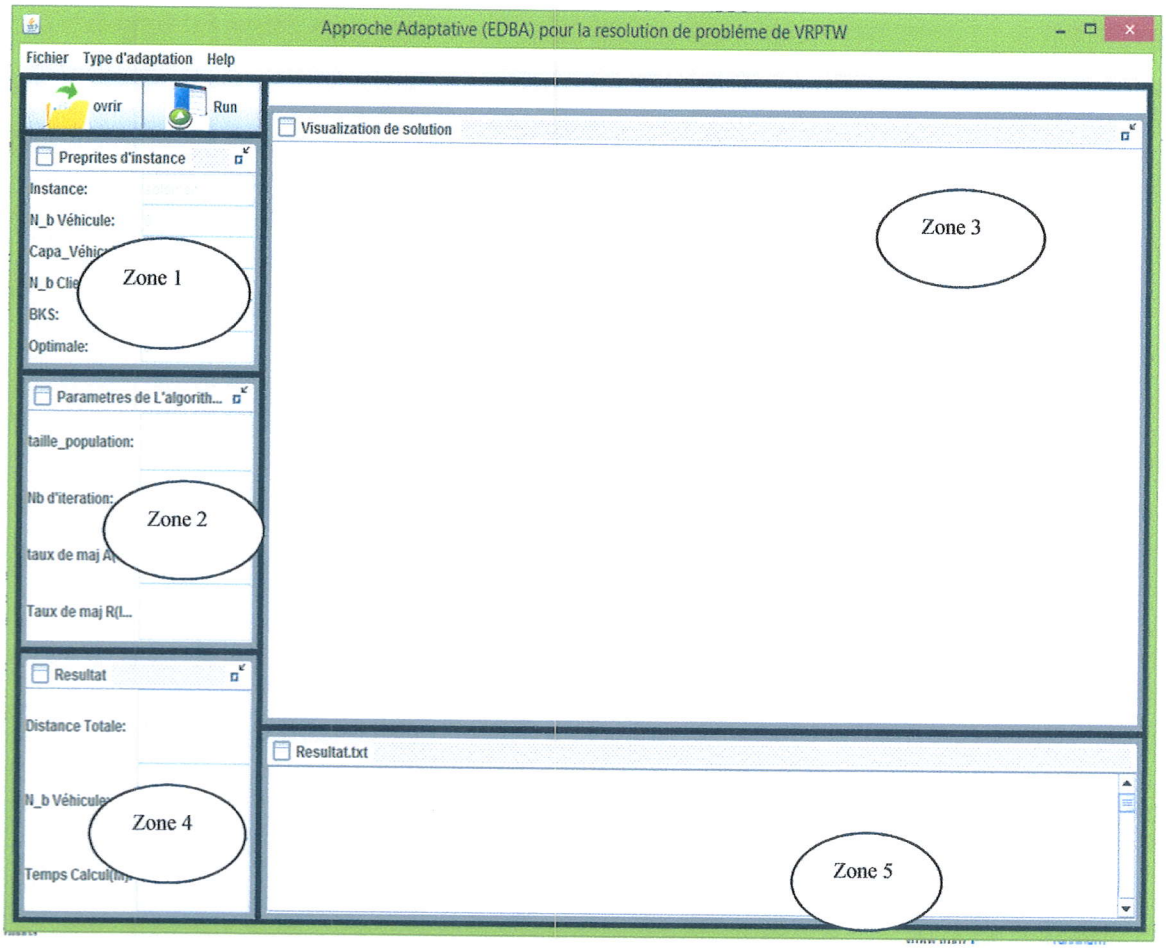
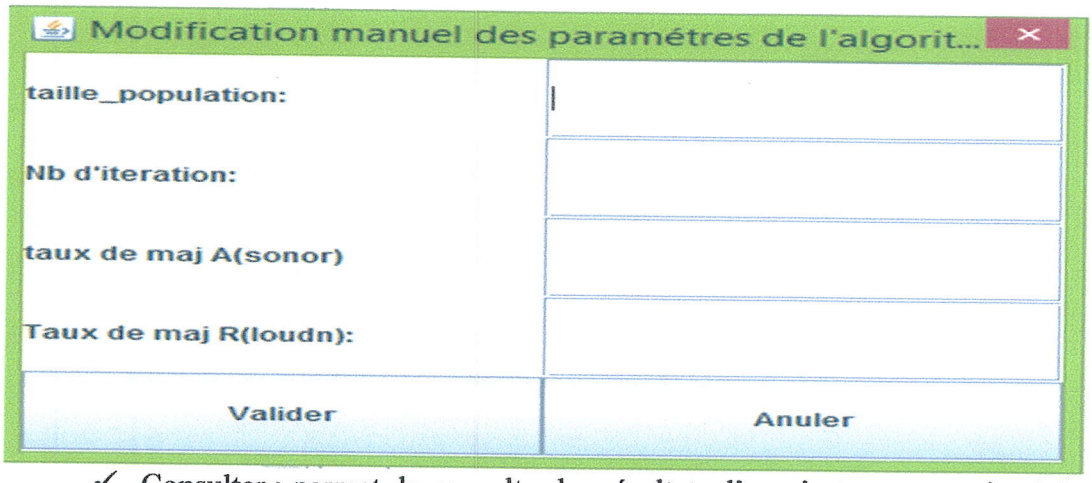


Figure 4.3 L'interface de l'application.

Les composants principaux de cette interface sont :

- Fichier : Il contient les opérations suivantes :



- ✓ Consulter : permet de consulter les résultats d'une instance enregistré dans un fichier texte.
- ✓ Quitter : permet de fermer complètement l'application.

- Type d'adaptation : elle contient les opérations suivantes :
  - ✓ Manuel : permet d'afficher une petite fenêtre (la figure 4.4) pour régler les paramètres de l'algorithme manuellement.
  - ✓ Automatique : permet de régler les paramètres de l'algorithme d'une manière automatique (Technique d'adaptation).

Figure 4.4 La fenêtre de réglage des paramètres de l'algorithme EDBA.

- Ouvrir : Afficher une fenêtre (la figure 4.5) qui permet de charger les données d'une instance située dans n'importe quel endroit sur l'ordinateur.

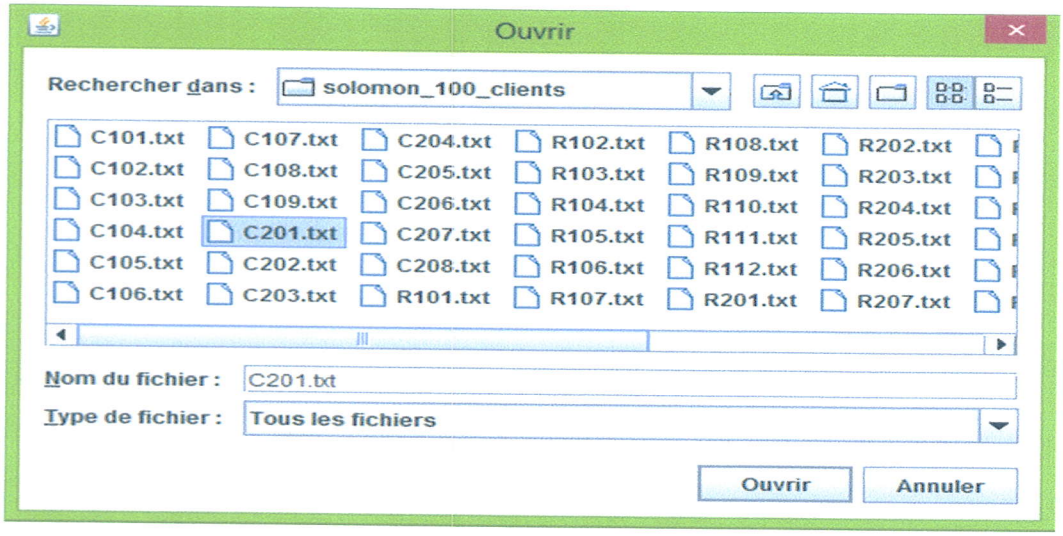


Figure 4.5 Chargement d'une instance de problème.

Après le chargement des données d'un problème la fenêtre (figure 4.5) est apparue.

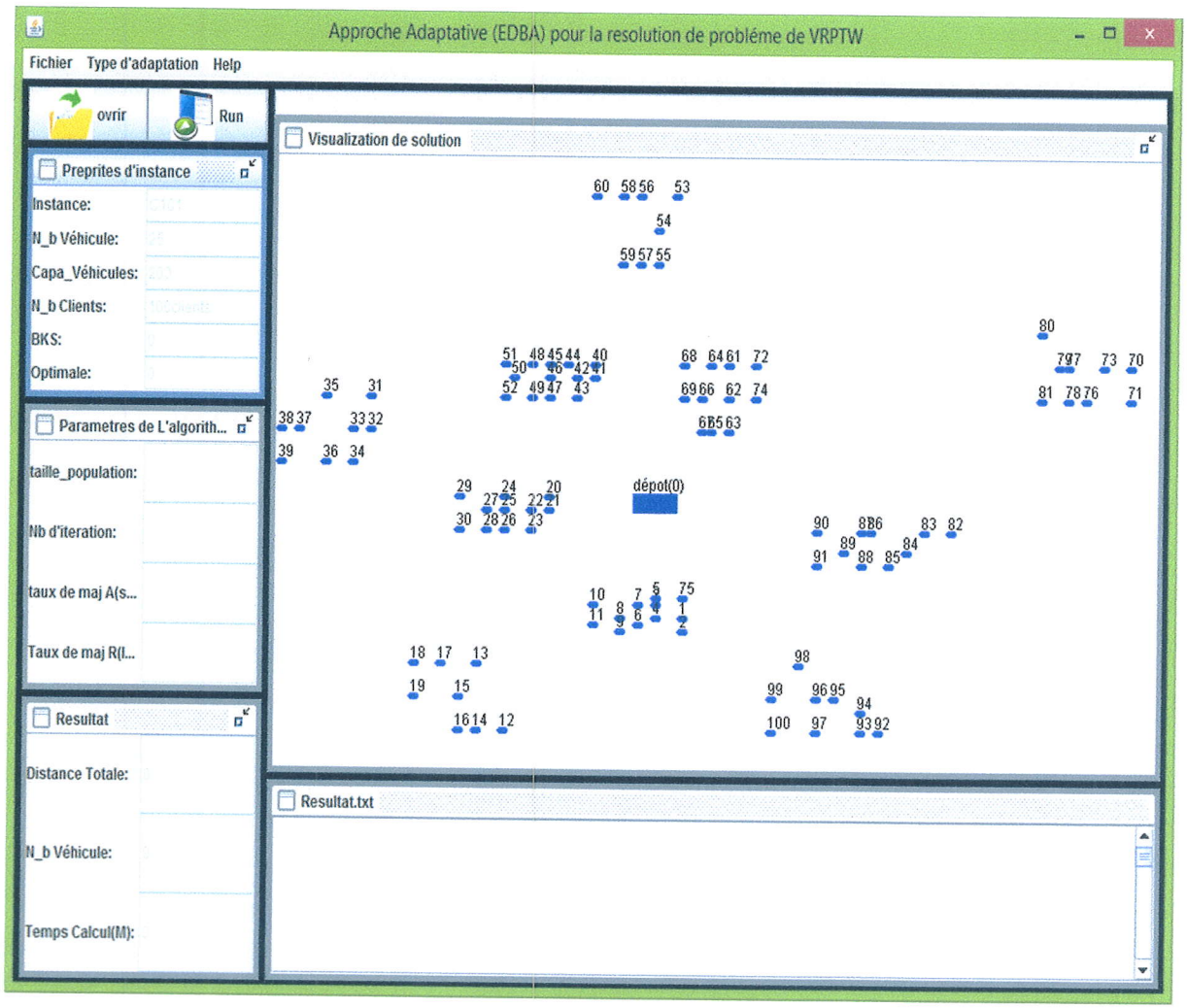


Figure 4.6 L'interface après chargement d'une instance.

- Run : concernant l'exécution de l'algorithme EDBA sur l'instance sélectionné.

Au lancement de l'exécution d'une instance chargée dans l'application, les propriétés de la solution finale sont affichées dans les zones (zone 1, zone 2, zone 3, zone 4, zone 5).

- ✓ Propriétés d'instances (zone 1) : concerne l'affichage textuelle des Propriétés d'une instance chargée dans l'application (BKS : meilleures solutions dans littérature, nombre de véhicule, nom de l'instance, nom d'instance, capacité de véhicules).
- ✓ Paramètres de l'algorithme (zone 2) : concerne l'affichage textuelle des paramètres d'algorithme EDBA qui sont génères par la technique d'adaptation.
- ✓ Visualisation de solution (zone 3) : concerne l'affichage graphique d'une solution finale d'une instance chargée.

- ✓ Résultat (zone 4) : permet d'afficher la distance totale et le nombre de véhicules d'une solution finale.
- ✓ Résultat.txt (zone 5) : concerne l'affichage textuelle d'une solution finale d'une instance chargée (affichage des tournées).

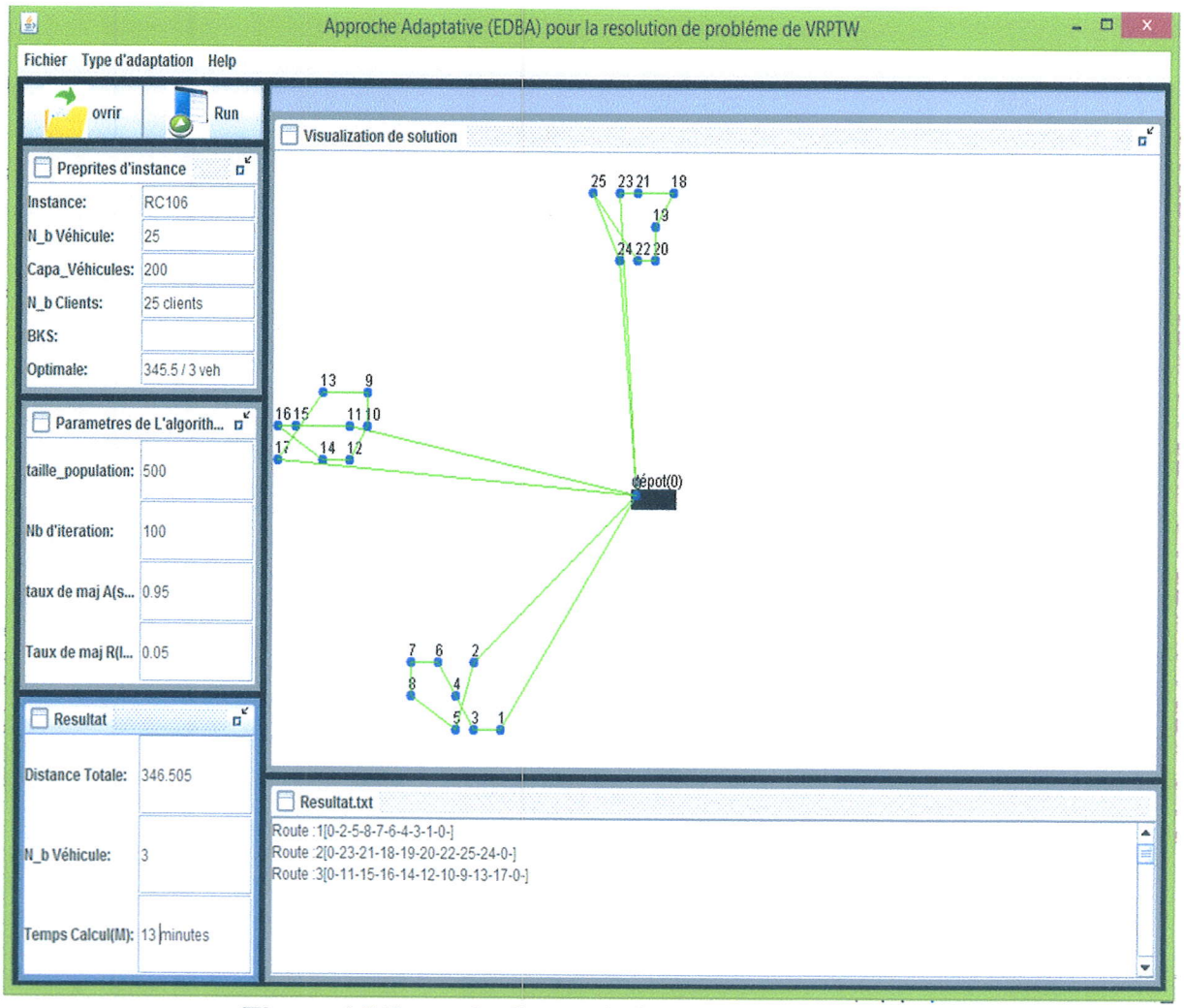


Figure 4.7 Exemple d'une solution d'une instance.

4.6 Résultats et étude comparative

Dans cette partie, nous comparons les résultats obtenus pour le jeu de données utilisé à l'aide de l'opérateur GAP qui calcule l'écart entre deux solution, et qui est défini comme suit :

$$GAP_{Approche1-Approche2} = \frac{Cost_{Approche1} - Cost_{Approche2}}{Cost_{Approche\,2}} \times 100\ \% \qquad (4.1)$$

Les résultats sont résumés dans les tableaux 4.3 et 4.7, pour le tableau 4.3 la première colonne représente le nom de l'instance la deuxième colonne (OS) représente les résultats optimales, les résultats obtenus par EDBA sont présentés dans la troisième colonne (EDBA)

La dernière colonne représente les écarts entre les solutions optimales et les solutions trouvées par EDBA. Pour le tableau 4.7 la première colonne représente le nom de l'instance la deuxième colonne (BKS) représente les meilleures résultats dans la littérature, les résultats obtenus par EDBA sont présentés dans la troisième colonne (EDBA), la dernière colonne représente les écarts entre les meilleures résultats dans la littérature et les résultats trouvées par EDBA.

Nous commençons les expériences en comparant les résultats de notre approche avec les résultats optimaux pour les instances de petites tailles (25 clients et 50 clients). Les résultats optimaux sont disponibles sur le site (http://w.cba.neu.edu/~msolomon/problems.html).

4.6.1 Comparaison résultats obtenus par « EDBA » avec les solutions optimales pour les instances de petites tailles.

A partir de tableau 4.4, nous remarquons que EDBA donne des résultats proches de l'optimal pour les trois types des instances de petites taille (25 et 50 clients) avec un écart moyen (de 1.38 %), pour les instances de 25 clients. Généralement EDBA donne des bons résultats pour les instances de petites tailles. Cela est dû au fait que l'espace de recherche est réduit.

- ✓ **Instances de 25 clients**

Le tableau 4.1 présente les résultats (la distance totale parcourue et le nombre de véhicule) obtenus pour les instances de 25 clients en appliquant l'approche EDBA.

Instances	Distance totale	Nb véhicule
C102_25	190.73	3
C201_25	215.54	2
R107_25	425.26	4
R208_25	329.33	1
RC107_25	298.94	3
RC201_25	362.20	3

Table 4.1 Résultats obtenus par l'approche EDBA pour les instances de 25 clients.

- ✓ **Instances de 50 clients**

Le tableau 4.2 présente les résultats (la distance totale parcourue et le nombre de véhicule) obtenus pour les instances de 50 clients en appliquant l'approche EDBA.

Instances	Distance totale	Nb véhicule
C102_50	362.17	5
C103_50	362.17	5
R101_50	1095.99	14
R210_50	702.74	3
RC108_50	599.17	6
RC201_50	686.31	5

Table 4.2 Résultats obtenus par l'approche EDBA pour les instances de 50 clients.

- **Solutions optimales**

Le tableau 4.3 présente les résultats optimaux pour les instances de 25 et 50 clients.

Instances	Distance totale	Nb véhicule	Auteurs
C102_25	190.3	3	[76]
C201_25	214.7	2	[77] + [79]
R107_25	424.3	4	[76]
R208_25	328.2	1	[80] + [78]
RC107_25	298.3	3	[76]
RC201_25	360.2	3	[77] + [79]
C102_50	361.4	5	[76]
C103_50	361.4	5	[76]
R101_50	1044.0	12	[76]
R210_50	645.6	4	[80] + [[78]
RC108_50	598.1	6	[76]
RC201_50	684.8	5	[79] + [81]

Table 4.3 Solutions optimales pour les instances de 50 et 25 clients.

- **EDBA Vs Solutions optimales (instance de 25 et 50 clients)**

Instances	OS		EDBA		Différence	
	Distance	*Véhicule*	*Distance*	*Véhicule*	*Δ(Distance)*	*GAP(%)*
C102_25	190.3	3	190.73	3	0.42	0.22
C201_25	214.7	2	215.54	2	0.84	0.39
R107_25	424.3	4	425.26	4	0.96	0.22
R208_25	328.2	1	329.33	1	1.13	0.34
RC107_25	298.3	3	298.94	3	0.63	0.21
RC201_25	360.2	3	362.20	3	2	0.55
C102_50	361.4	5	362.17	5	0.77	0.21

C103_50	361.4	5	362.17	5	0.77	0.21
R101_50	1044.0	12	1095.99	14	51.99	4.97
R210_50	645.6	4	702.74	3	57.14	8.85
RC108_50	598.10	6	599.17	6	1.06	0.17
RC201_50	684.8	5	686.31	5	1.51	0.22
Moyenne						**1.38**

Table 4.4 EDBA Vs Solutions optimales.

4.6.2 Comparaison résultats obtenus par « EDBA » avec les meilleurs résultats dans la littérature BKS pour les instances de grande taille

A partir de tableau 4.7, nous remarquons que, dans les instances de type (C1, C2) où les clients sont regroupés, les résultats obtenus sont proches des BKS avec un écart de (0.14 %-5.75%) pour les groupes C1, C2 respectivement. Par contre, les résultats obtenus pour les groupes d'instances (R et RC) sont moins bons.

✓ **Instances de 100 clients**

Le tableau 4.5 présente les résultats (la distance totale parcourue et le nombre de véhicule) obtenus pour les instances de 100 clients en appliquant l'approche EDBA.

Instance	Distance totale	Nb véhicule
C102_100	830.15	10
C201_100	625.70	3
R101_100	1846.17	22
R208_100	829.44	3
RC107_100	1395.01	13
RC205_100	1485.16	6

Table 4.5 Résultats obtenus par l'approche EDBA pour les instances de 100 clients.

➢ **Meilleurs résultats dans littérature** (BKS)

Le tableau 4.6 présente les meilleurs résultats obtenus dans la littérature (distance totale parcourue et le nombre de véhicule) pour les instances de 100 clients.

Instance	Distance totale	Nb véhicule	Auteurs
C102_100	828.94	10	[82]
C201_100	591.56	3	[82]
R101_100	1645.79	19	[83]
R208_100	726.75	2	[84]
RC107_100	1230.48	11	[85]
RC205_100	1297.19	4	[84]

Table 4.6 Meilleurs résultats dans la littérature pour les instances de 100 clients.

- **EDBA Vs meilleures résultats dans la littérature (BKS) (instance de 100 clients)**

Instances	BKS		EDBA		Différence	
	Distance	*Véhicule*	*Distance*	*Véhicule*	*Δ(Distance)*	*GAP (%)*
C102_100	828.94	10	830.15	10	1.20	0.14
C201_100	591.56	3	625.7	3	34.14	5.77
R101_100	1645.79	19	1846.17	22	200.38	12.18
R208_100	726.75	2	829.44	3	102.69	14.13
RC107_100	1230.48	11	1395.02	13	164.54	13.37
RC205_100	1297.19	4	1485.16	6	187.97	12.65
Moyenne						**9.70**

Table 4.7 EDBA Vs meilleures résultats dans littérature (BKS)

En résumé et à partir des résultats, nous remarquons que l'approche proposée a donné des résultats proches de l'optimal dans les instances de petites tailles (25 et 50 clients). Pour les grandes instances, l'écart entre les solutions est moyen.

4.7 Conclusion

Dans ce mémoire, nous nous intéressons au problème de VRPTW, qui est une variante très étudiée. Ce problème est NP-difficile, nous l'avons résolu à l'aide d'une nouvelle métaheuristique nommée BA (Bat Algorithm). Cette dernière étant conçue pour des problèmes continus, nous avons adapté une version discrète améliorée, nommée (EDBA). Nous avons amélioré l'approche proposée par l'intégration de l'algorithme de recuit simulé pour améliorer la qualité de la solution, à chaque itération.

Nous ajustons les paramètres de l'algorithme automatiquement en utilisant la méthode de descente pour chaque instance de problème. Nous avons utilisé différentes instances de problèmes dérivés des benchmarks de Solomon pour comparer EDBA avec BKS. Les résultats expérimentaux ont démontré l'efficacité de l'algorithme proposé (EDBA). Ainsi, nous croyons qu'EDBA représente une bonne approche pour résoudre le VRPTW.

Conclusion générale

Le travail élaboré dans ce mémoire porte essentiellement sur les problèmes des tournées de véhicules et précisément sur les problèmes de tournées de véhicules avec fenêtre de temps, qui permettent la modélisation de nombreux phénomènes de la logistique et de l'économie.

L'objectif était de prospecter les voies d'adaptation dans les métaheuristiques pour résoudre le problème cité. Les deux premiers chapitres ont été consacrés intégralement aux définitions et à la présentation de l'état de l'art du VRP et de ses variantes et ces méthodes de résolution. Deux types de méthodes de résolution des problèmes NP-DUR tel que le VRPTW, ont été abordées : les méthodes exactes qui assurent l'optimalité de la solution mais pour des problèmes de petites instances (taille limitée), et les approches heuristiques et métaheuristiques qui fournissent des solutions appréciables en un temps raisonnable a des problèmes de n'importe quelle instance.

Ensuite, après avoir décrire le principe et les caractéristiques de l'algorithme de chauve-souris (BA), nous avons proposé notre approche de résolution du problème de VRPTW, cette approche est basée principalement sur l'algorithme BA amélioré à base de méthode de recuit simulé en utilisant l'aspect d'adaptation. Cet aspect a été utilisé pour le réglage automatique des paramètres de la métaheuristique utilisée. Ce réglage est très important pour la qualité des résultats obtenus.

L'approche proposée passe par une phase d'adaptation des paramètres aux instances de problème traité. Nous avons testé notre approche 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, nous avons testé notre approche sur 18 instance (6 instance de 25 clients, 6 instance de 50 clients et 6 instance de 100 clients). Les résultats sont comparés avec les résultats optimaux pour les instances de petites tailles (25 et 50 clients), pour les instances de grande taille (100 clients) nous avons comparés nos résultats avec les meilleurs résultats dans la littérature.

Les tests effectués sur ces instances ont montré que notre approche donnent des résultats proches de l'optimal pour les instances de petite taille (25 et 50 clients).

Pour les instances de grande taille (100 clients), nous arrivons à des solutions proches de celles des BKS pour le cas des problèmes de catégorie C (qui sont générées de façon à assurer le regroupement des clients en sous-ensembles ou en clusters).

Pour le cas des problèmes des catégories RC et R notre approche donne des solutions moins performant que celles des BKS.

Il resterait désormais d'établir des analyses expérimentales plus poussées, permettant de valider l'efficacité de la méthode adaptative proposée.

Bibliographie

[1] LAYEB, Abdesslem. Thèse présentée à la faculté des Sciences de l'Ingénieur en vue de L'obtention du grade de Docteur en science en Informatique . Uniersité Mentouri de Constantine : Utilisation des Approches d'Optimisation Combinatoire pour la Vérification des Applications Temps Réel, 27/05/2010.

[2] Gherbouj, Amira. «Méthodes de résolution de de problèmes difficiles,académiques. Thèse Pour l'obtention du diplôme de Doctorat 3ème cycle LMD Spécialité Informatique. Université de Constantine2, 2013.

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

[4] Thèse en vue de l'obtention du Doctorat . université de toulouse.''De l'ordonnancement déterministe à l'ordonnancement distribué sous incertitudes''.Samia,OURARI. Le 28 janvier 2011.

[5] Kammarti, Ryan. « These Approches Evolutionnist Pour La Resolution Du 1-PDPTW Statique Et Dynamique. » Lille, 2006.

[6] Prodhon, Caroline. Thèse de doctorat de l'Université de Technologie de Troyes. Le Problème de Localisation-Routage, 2006.

[7] Fred Glover et Manuel Laguna, Tabu Search.Kluwer Academic Publishers , Norwelle , MA, USA, 1997. (cité en page 17)

[8] Helena R Lourenço, Olivier G Martin et Thomas Stutzle.Iterated local search.Kluwer Academic Publishers, 2002.

[9] N Mladenovic et P Hasen. Variable neighborhood search. Computers & Operations Research, vol. 24, no 11, 1997

[10] Thomas A Feo et Mauricio G C Resende. A Probabilistic heuristic for a

computationally difficult set covering problem. Operations Research Letters, vol. 8, no. 2, 1989.

[11] Thomas A Feo et Mauricio G C Resende. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, vol. 6, 1995.

[12] Alberto Colorni, Marco Dorigo et Vittorio Maniezzo. Distributed Optimization by Ant Colonie. In European Conference on Artificial life,1991.

[13] James Kennedy et Rusell G. Eberhart. particle swarm optimization. In proceedings of the IEEE International Conference on Neural Networks, 1995.

[14] Melanie Mitchell. AN introduction to genetic algorithems.MIT Press, Combridge, MA, USA, 1998.

[15] Wolfgang Banzhaf, Frank D. Francone, Robert E. Keller et Peter Nordin.Genetic programming : An introduction : on the automatic evolution of computer programs and its application. Morgan Kaufmann Publishers Inc. San Francisco, CA,USA , 1998.

[16] Xianshun Chen, Yew-Soon Ong, Meng-Hiot Lim et Kay Chen Tan. A Multi-Facet Survey on Memetic Computation. IEEE Trans. Evolutionary Computation, vol. 15, no. 5, 2011.

[17] Refael Marti, Manuel Laguna et Fred Glover. Principales of scatter search. European Journal of Operational Research, vol. 169, no.2, 2006.

[18] Duc Troung pham, Marco Castellani et le Thi Hoai An. Nature-Inspired Intelligent Optimisation Using the Bees Algorithm. T.Computational Collective Intelligence, vol. 13,2014.

[19] Xin-She Yang. Nature-inspired metaheuristic algorithms : Second edition. Luniver Prees, 2010.

[20] X in-She Yang et Suash Deb. Cuckoo Search via Lévy Flights. In NaBIC, pages 210-214. IEEE.

[21] L. N. de castro et F. J. Von Zuben. learning and Optimization Using the Clonol Selection Principale. Trans.Evol. Comp, vol.6, no. 3,juin 2002.

[22] Zong Woo Geem, Joong Hoon Kim et G V Loganathan. A new heuristic optimization algorithm: harmony Search. Simulation. vol. 76, no. 2, 2001.

[23] Stefan Boettcher et Allon G. percus. Extremal Optimization : Methods derived from Co-Evolution. In Wolfgang Banzhaf, jason Daida,Agoston E. Eiben, Max H. Grzon, Vasant Hasant Honavar, Mark Jakiela et Robert E. Smith, editeurs, Proceedings of the Genetic and Evolutionary Computation Conference, volume 1,Orlando, Floida, USA, 13-17 july 1999. Morgan Koufmann.

[24] Chris Voudouris et Edward Tsang. Guided local search. Rapport technique, European Journal of Operational Research, 1995.

[25] Berghida Meryem. Systèmes Informatiques. Stratégies adaptatives et coopératives pour la résolution de problèmes de tournées de véhicules avec collecte et livraison, le 25/04/2015.

[26] Raksmey PHAN, Une stratégie hybride pour le problème de tournées de véhicules avec livraisons et collectes. Laboratoire d'Informatique, de Modélisation et d'optimisation des systèmes, septembre 2009.

[27] Cesar Rego et Catrine Roucairol, Le Problème de Tournéées de Véhicules : étude et résolution approchée. INRIA, féevrier 1994.

[28] Nicos Christo_des. The vehicle routing problem. RAIRO - Operations Research -Recherche Opérationnelle, vol. 10, no. V1, 1976.

[29] Paolo Toth et Daniele Vigo, editeurs. The vehicle routing problem.Society for Industrial and Applied Mathematics, Philadelphia,PA, USA, 2001.

[30] C Rego, C Roucairol et Institut National de recherche en informatiqueet en automatique. Le problème de tournées de véhicules :étude et résolution approchée. Rapports de recherche. Institut national de recherche en informatique et en automatique, 1994.

[31] 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, 2003.

[32] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon et F. Soumis. The Vehicle Routing Problem. chapitre VRP with Time Windows. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[33] 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.

[34] Paolo Toth et Daniele Vigo, editeurs. The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[35] R. Mechti, "Tournée de véhicules à la demande: problèmes et méthodes". Technical report, Laboratoire PRiSM, Université de Versaille-St. Quentin, France, 1995.

[36] Yannis Marinakis et Magdalene Marinaki. A Bumble Bees Mating Optimization algorithm for the Open Vehicle Routing Problem.Swarm and Evolutionary Computation, no. 0, 2013.

[37] Shiwei Yu, Chang Ding et Kejun Zhu. A hybrid GA-TS algorithmfor open vehicle routing optimization of coal mines material. ExpertSystems with Applications, vol. 38, no. 8, 2011.

[38] D Sariklis et S Powell. A heuristic method for the open vehicle routing problem. Journal of the Operational Research Society, vol. 51,no. 5, 2000.

[39] G Laporte. The vehicle routing problem : An overview of exact and approximate

algorithms. European Journal of Operational Research, vol. 59, no. 3, 1992.

[40] Sophie Parragh, Karl Doerner et Richard Hartl. A survey on pickup and delivery problems. Journal für Betriebswirtschaft, vol. 58,no. 2, 2008.

[41] Fan Wang, Yi Tao et Ning Shi. A Survey on Vehicle Routing Problem with Loading Constraints. In Lean Yu, Kin Keung Lai et S KMishra, editeurs, CSO (2). IEEE Computer Society, 2009.

[42] Victor Pillac, Michel Gendreau, Christelle Guéret et Andrés Medaglia. A review of dynamic vehicle routing problems. EuropeanJournal of Operational Research, vol. 225, no. 1, 2013.

[43] 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, 2012.

[44] Suresh Nanda Kumar; Ramasamy Panneer selvam. A Survey on the Vehicle Routing Problem and Its Variants. Intelligent Information Management, vol. 4, no. 3, 2012.

[45] 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, 2014.

[46] 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, 2008.

[47] Irina Gribkovskaia, Ø yvind Halskau Sr., Gilbert Laporte et Martin Vl£ek. General Solutions to the single vehicle routing problem with pickups and deliveries. European Journal of Operational Research, vol. 180, no. 2, 2007.

[48] Rita Macedo, Claudio Alves, JM Valério de Carvalho, François Clautiaux et Saïd Hana. 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, 2011.

[49] 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, 2011.

[50] Russell Bent et Pascal Van Hentenryck. A two-stage hybrid algorithmfor pickup and 33, delivery vehicle routing problems with time windows. Computers & Operations Research, vol. no. 4, 2006.

[51] 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, Springer, 2008.

[52] 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, 2009.

[53] Anand Subramanian, Eduardo Uchoa et Luiz SatoruOchi. A hybrid algorithm for a class of vehicle routing problems. Computers& Operations Research, vol. 40, no. 10, 2013.

[54] Shuguang Liu. A hybrid population heuristic for the heterogeneous vehicle routing problems. Transportation Research Part E: Logistic sand Transportation Review, vol. 54, no. 0, 2013.

[55] 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.

[56] 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 of the Computing Lab at the Center for the Study of Complex Systems University of Michigan Rick Riolo Research Scientist. Perspective son Adaptation in Natural and Artificial Systems. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, USA, 2005.

[57] David H. Wolpert et William G. Macready. No free lunch theorems for optimization. IEEE TRANSACTIONS ON EVOLUTIO-NARY COMPUTATION, vol. 1, no. 1, 1997.

[58] Janez Brest, Ales Zamuda et Borko Bokovié, Adaptation in the Diferential Evolution. In Iztok Fister et Iztok Fister Jr., editeurs, Adaptation and Hybridization in Computational Intelligence SE - 2, volume 18 of Adaptation, Learning, and Optimization, Springer International Publishing, 2015.

[59] 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. Springer International Publishing, 2015.

[60] Giuseppe A. 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. Springer Inter-national Publishing, 2015.

[61] X. S. Yang, "A New Metaheuristic Bat-Inspired Algorithm", Nature Inspired Cooperative Strategies for Optimization, 2010.

[62] D. R. Griffin, "Listening in the dark", Yale Univ. Press, New York, 1958.

[63] J. R, Speak man, P. A. Racey, "The cost of being a bat", Nature V 350, 1991.

[64] Pappalardo E, Pardalos P, Stracquadanio G (2013) Mathematical optimization. Optimization approaches for solving string selection problems, Springer Briefs in

Optimization Springer, New York.

[65] Du D-Z, Pardalos PM (1999) Handbook of combinatorial optimization: supplement, vol1. Springer, Berlin.

[66] Yang X-S (2010) A new metaheuristic bat-inspired algorithm. Nature inspired Cooperative strategies for optimization (NICSO2010). Springer, Berlin.

[67] Yang X-S, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering Optimization. Eng Comput 29(5).

[68] Yang X-S, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4).

[69] Ouaarab A, Ahiod B, Yang X-S (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput A pp l24 (7–8).

[70] Kenndy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. pp 1942–1948

[71] Nakamura RY, Pereira LA, Costa K, Rodrigues D, Papa JP, Yang X-S (2012) BBA : A binary bat algorithm for feature selection. In: Graphics, patterns and images (SIBGRAPI), 2012 25th SIBGRAPI Conference on IEEE.

[72] Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm Algorithm. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation. 1997 IEEE International Conférence on, IEEE.

[73] E. Bonomi et J.-L. Lutton ; le recuit simulé pour la science ; numéro 129, Juillet 1988.

[74] Marshall L. Fisher et Ramchandran Jaikumar. A generalized assignment heuristic for Vehicle routing. Networks, vol. 11, no. 2, pages 109124, 1981.

[75] M.W.P. Save l sbergh, "Local Search for Routing Problems with Tirne Windows".

Annals of Operations Research 4, 1995

[76] KDMSS - N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Science, Soumis, "2-Path Cuts for the Vehicle Routing Problem with Time Windows," Transportation Vol. 33 (1), (1999).

[77] CR - W. Cook and J. L. Rich, "A parallel cutting plane algorithm for the vehicle routing problem with time windows," Working Paper, Computational and Applied Mathematics, Rice University, Houston, TX, 1999.

[78] C - A. Chabrier, "Vehicle Routing Problem with Elementary Shortest Path based Column Generation." Forthcoming in: Computers and Operations Research (2005).

[79] L - J. Larsen. "Parallelization of the vehicle routing problem with time windows." Ph.D. Thesis IMM-PHD-1999-62, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[80] IV - S. Irnich and D. Villeneuve, "The shortest path problem with k-cycle elimination ($k \geq 3$): Improving a branch-and-price algorithm for the VRPTW." Forthcoming in: INFORMS Journal of Computing (2005).

[81] KLM - B. Kallehauge, J. Larsen, and O.B.G. Madsen. "Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results." Internal report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.

[82] RT - Y. Rochat and E.D. Taillard, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," Journal of Heuristics 1, 147-167, (1995).

[83] H - J. Homberger, "Verteilt-parallele Metaheuristiken zur Tourenplanung," Gaber, Wiesbaden (2000).

[84] M - D. Master, "An Evolutionary Strategies Algorithm for Large Scale Vehicle Routing Problem with Capacitate and Time Windows Restrictions," Working Paper,

Institute of Evolution, University of Haifa, Israel (2002).

[85] S97 - P. Shaw, "A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems," Working Paper, University of Strathclyde, Glasgow, Scotland, 1997.

[86] BVH - R. Bent and P. Van Hentenryck, "A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows," Technical Report CS-01-06, Department of Computer Science, Brown University, 2001.

Résumé

Le VRPTW (Problème de tournées de véhicules avec fenêtres de temps) est constitué d'une flotte de véhicules qui doit servir un ensemble de clients dispersés géographiquement, chacun ayant une demande connue et un intervalle de temps, durant lequel il peut être servi. Dans ce mémoire, nous présentons une nouvelle approche adaptative (EDBA) pour la résolution de ce problème. Cette approche basée sur l'algorithme discret de chauve-souris amélioré par la méthode de recuit simulé.

Les résultats obtenus dans l'expérimentation, pour les petites et grandes instances, montrent l'efficacité de l'approche proposée.

Mots clés : Métaheuristiques adaptatives, Recuit simulé, problème de tournées de véhicules, fenêtres de temps, algorithme de chauves-souris, VRPTW.

Abstract

The VRPTW (Vehicle Routing Problem with Time Windows) consists of a fleet of vehicles which must serve a set of geographically dispersed customers, each with a known demand and a time window, during which it can be served. In this paper, we present a new adaptive approach (EDBA) to solve this problem. This approach named Discrete Bat Algorithm is improved by simulated annealing method.

The results obtained in experimentation, for small and large instances, show the effectiveness of the proposed approach.

Keywords: adaptive metaheuristics, Simulated Annealing, Vehicle Routing Problem, Time Windows, Bat Algorithm, VRPTW.