SULTAN MOULAY SLIMANE UNIVERSITY

DOCTORAL THESIS

Vehicle Routing Problem with Time Windows: From mathematical models to computing science Tools

Author:
Mehdi NASRI

Supervisor:
Dr. Abdelmoutalib METRANE
Dr. Imad HAFIDI

*A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Mathematics & Computer Sciences*

in the

LIPIM Laboratory
National School of Applied Sciences of KHOURIBGA

Declaration of Authorship

I, Mehdi NASRI, declare that this thesis titled, "Vehicle Routing Problem with Time Windows: From mathematical models to computing science Tools" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

SULTAN MOULAY SLIMANE UNIVERSITY

Abstract

National School of Applied Sciences
National School of Applied Sciences of KHOURIBGA

Doctor of Mathematics & Computer Sciences

Vehicle Routing Problem with Time Windows: From mathematical models to computing science Tools

by Mehdi NASRI

This dissertation focuses on vehicle routing problems, subject of massive investigations in operations research. we address a specific variant of the problem called in the literature vehicle routing problem with time windows (VRPTW), when the solution has to obey several other constraints, such as the consideration of travel, service, and waiting times together with time-window restrictions. An interesting topic on solving this problem consists in considering parameters affected by uncertainty, making the problem more realistic. Thus, we present robust optimization methodology for the VRPTW under uncertainty in which we outline a specific robust VRPTW model, depending on the source of the uncertainty (service times, travel times), VRPTW formulation, and correlation between uncertain coefficients.

The aim of this research is to develop new algorithms for the considered problems, investigate their performance and compare them with the literature approaches. Two cases are carried out. The first case studies the deterministic Vehicle Routing Problem with Time Windows (VRPTW). We apply the Adaptive Large Neighborhood Search algorithm (ALNS) to deal with the problem. Then we suggest three different strategies to improve the ALNS process in terms of objective function and execution time. The second case studies the Robust Vehicle Routing Problem with Time Windows (RVRPTW). Both sequential and parallel methods are proposed to deal with this problem. We report that our robust methodologies to find routing plans give rise to encouraging solutions which protect from the violation of time windows for different scenarios.

SULTAN MOULAY SLIMANE UNIVERSITY

Résumé

National School of Applied Sciences
National School of Applied Sciences of KHOURIBGA

Doctor of Mathematics & Computer Sciences

Vehicle Routing Problem with Time Windows: From mathematical models to computing science Tools

by Mehdi NASRI

Cette thèse s'intéresse à la résolution de deux variantes du problème de tournées de véhicules, qui intègrent des contraintes ou des objectifs de maîtrise des risques : le problème de tournées de véhicules avec contraintes de temps déterministe (VRPTW), et le problème de tournées de véhicules avec contraintes de temps sous incertitudes de temps de service et de déplacement (RVRPTW). Les applications ciblées sont du domaine de la logistique et du transport.

Le problème de tournées de véhicules avec fenêtres de temps a fait l'objet de la majeure partie des travaux de la recherche opérationnelle. Il consiste à définir un ensemble de routes de véhicules de coût total minimal, afin de collecter ou de livrer des marchandises à des clients en respectant des fenêtres de temps. Ces clients sont généralement représentés par des noeuds sur un réseau, et doivent être visités une et une seule fois par un véhicule, leur demande doit être satisfaite par cette seule visite. Le VRPTW a été très étudié et de nombreux algorithmes de résolution ont été proposés.

Le mémoire se découpe en quatre parties. Le chapitre 1 est une introduction intégrant une revue de littérature qui recense les principaux travaux sur les problèmes de tournées de véhicules ainsi que les méthodes de résolution, il définit également le problème étudié dans sa version déterministe et robuste. Le chapitre 2 est une définition générale du problème de VRPTW ainsi qu'une présentation de l'algorithme de la recherche adaptive à voisinage large (ALNS) pour le VRPTW qui sera modifié dans le cadre de cette recherche.

La recherche adaptive à voisinage large (ALNS) de Pisinger and Ropke, 2007, exploite les bénéfices des voisinages variés, à base des opérateurs de type destruction/reconstruction de Shaw, 1998, en adaptant la fréquence d'utilisation de chaque opérateur en fonction de leur performance dans l'historique de la recherche. Autrement dit, ALNS utilise de nombreux opérateurs de destruction et de reconstruction afin de diversifier la recherche. Elle est nommée adaptative puisqu'elle rajoute une évaluation de chaque opérateur de destruction ou de reconstruction, basée sur ses performances aux itérations précédentes. Ainsi, chaque opérateur reçoit un score qui est régulièrement mis à jour. La probabilité de choisir un opérateur à l'itération courante dépend de ce score.

En effet, les modifications apportées à l'ALNS ont pour objectif l'amélioration de la fonction objectif ainsi que la diminution du temps d'exécution. Pour cette raison, on propose dans la section 2.3 deux techniques pour améliorer la qualité de la solution obtenue. Le principal défi de la première méthode est de mettre en évidence l'intensification par rapport à la diversification dans le processus de recherche heuristique. Dans ce contexte, nous incorporons la fonction de choix proposé par Drake, Özcan, and Burke, 2012 dans le processus de la méthode ALNS comme étant un critère de sélection de l'opérateur adéquat à chaque itération. Par conséquent, plusieurs méthodes de destruction / réparation sont combinées pour explorer plusieurs voisinages au sein d'une même recherche.

Lors de la deuxième alternative, nous proposons une nouvelle approche qui s'inscrit dans la classe des algorithmes Cluster first - Route second pour traiter le problème VRPTW. Cette stratégie comme son nom l'indique, se compose de deux phases :

- La phase Cluster : établir un certain nombre de clusters (k clusters lorsque k véhicules sont disponibles).
- La phase route : Nous relions tous les clients d'un même cluster par un seul tour, autrement, on résout un problème VRPTW sur chaque cluster.

La première phase vise à définir un ensemble de groupes faisables et rentables à l'aide l'algorithme k-Medoid en se basant sur une distance spatio-temporelle efficace qui est totalement appropriée à la nature du VRPTW, étant donné qu'il prend en compte à la fois les dimensions spatiales et temporelles du problème, alors que la deuxième phase est consacrée à la sélection des routes adéquates. En considérant que chaque cluster correspond à un sous-problème VRPTW spécifique, nous essayons de le résoudre en appliquant trois algorithmes de routage différents à savoir (La recherche adaptive à voisinage large (ALNS), la recherche à voisinage variable (VNS), et l'algorithme génétique (GA)) séparément afin de valider les résultats trouvés au premier niveau. Il est à noter que le choix du K-Medoid n'était pas arbitraire car il est plus robuste et il est plus flexible pour être utilisé avec toute mesure de similitude contrairement à d'autres techniques de partitionnement qui ne sont pas sensibles aux données bruyantes ou doivent être utilisées uniquement avec des distances cohérentes avec la moyenne (par exemple K-Means). La mesure de similarité utilisée dans l'algorithme K-Medoid a été comparée à certains algorithmes de partitionnement de la littérature tels que l'algorithme K-Means, et conduit à des solutions avec des coûts minimaux, quelles que soient les métaheuristiques utilisées pour le routage.

Dans la section 2.4, nous proposons une nouvelle approche de parallélisation de l'algorithme de recherche adaptive à voisinage large conçu pour résoudre le problème de tournées des véhicules avec des fenêtres de temps. Notre objectif est d'obtenir une solution en temps réel réduit sans compromettre la qualité de la solution spécialement pour les grandes instances. Notre principale contribution introduit une procédure pour la parallélisation de la méthode Greedy utilisée dans le bloc d'initialisation, ainsi que les opérateurs de destruction / réparation impliqués dans le processus de la méta-heuristique ALNS. Pour améliorer notre approche, nous utilisons un algorithme de clustering itératif comme prétraitement pour affecter les clients aux threads de calcul. Pour ceci, nous adoptons le K-Means comme prototype, ce qui n'est pas une méthode restrictive, nous pouvons adopter d'autres techniques telles que les K-Medoids ou le clustering spatial basé sur la densité d'applications, (voir par exemple Cömert et al., 2017). Les résultats sont élaborés de telle manière à trouver un compromis entre la qualité de la solution et le temps d'exécution.

Dans le chapitre 3, nous considérons une variante du problème de tournées de véhicules avec fenêtres de temps (VRPTW) ou les temps de service et les temps de trajet sont incertains et représentés par des ensembles discrets de valeurs incertaines. Notre contribution à tous les travaux antérieurs réside, tout d'abord, dans la modélisation du problème, en se basant sur l'approche de Bertsimas and Sim, 2003, le choix de la métaheuristique Adaptive Large Neighbourhood Search (ALNS) pour l'intégrer dans notre approche afin de traiter le VRPTW robuste. De plus, les résultats numériques ont été testés sur un ensemble de petites instances basées sur le benchmark de Solomon et de grandes instances du benchmark de Gehring & Homberger. Le problème étudié a généré différents scénarios et chaque scénario est réalisé en utilisant la meilleure méthode d'échantillonnage connue, qui est la version Metropolis de la simulation de Monte Carlo.

Afin d'améliorer cette approche et étudier l'effet de la parallélisation sur le temps d'exécution et la fonction objectif. Nous avons intégré la version parallèle e l'ALNS développée par Røpke, 2009, qui conduit à une réduction du temps de fonctionnement. De plus, nous avons utilisé notre version parallèle de l'algorithme Metropolis Monte Carlo pour générer toutes les réalisations possibles et pour transformer le problème sous incertitudes en un ensemble de sous-problèmes déterministes. Sur la base de la mise en œuvre efficace de Røpke, 2009, différentes combinaisons (séquentielles / parallèles) de l'algorithme de Monte Carlo et de l'ALNS sont effectuées. De cette façon, notre stratégie offre aux décideurs le choix de la combinaison en fonction de leurs préférences et de la situation.

Enfin, le quatrième chapitre n'est que la conclusion de cette recherche reprenant

les résultats importants ainsi que l'originalité des méthodes développées, et elle décrit quelques orientations pour les futures recherches.

Acknowledgements

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future.

I would like to express my deepest and sincere appreciation to my research supervisor, Dr. Abdelmoutalib METRANE, for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present the research works as clearly as possible.

Besides my supervisor, I would like to thank Dr. Imad HAFIDI. I could not have imagined having a better mentor for my Ph.D. study. He has provided his supervision, valuable guidance, continuous encouragement as well as given me extraordinary experiences through out the work.

I would like to take the opportunity to express my gratitude and to thank Dr. Abderrahim Azouani. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. His advices on research as well as on my career have been priceless.

So as to the rest of my family for their love and support. Their encouragement and understanding during the course of my dissertation make me pursue the advanced academic degree.

I am very honored to thank my jury for accepting to be a member of the PhD Thesis Committee:

- Dr. Said MELLIANI, Professor at FST - Beni Mellal
- Dr. Salah EL-HADAJ, Professor at ENCG - Marrakech
- Dr. Mohammed YOUSSFI, Professor at ENSET - Mohammedia
- Dr. Nizar EL HACHEMI, Professor at EMI - Rabat
- Dr. Mohammed BADAOUI, Professor at EST - Oujda

Contents

List of Figures

List of Tables

Dedicated To my parents, to all my family, to my professors, and to my friends.

Chapter 1

Introduction

1.1 Motivation

Logistics define the process of planning, implementing, and controlling procedures for the efficient and effective transportation and storage of goods including services, and related information from the point of origin to the point of consumption for the purpose of conforming to customer requirements. It has great importance in the economy, in industry and in environment protection. Motivated by those potential applications, logistics has attracted enormous attention from researchers and has became recognized as an area that was key to overall business success.

As such, a primary goal of logistics is optimization. Optimization is performed in cases of solving complex computing problems, routing, games, decision support, healthcare and many other instances related to problem solving and solutions are further formed to these problems as long as a perfectly optimized solution is obtained. There are numerous ways of optimizing a problem which are classified based on the type of the problem such as tools, strategies, standard methods, tests etc.

In fact, planning optimized routes is very important for companies who offer delivery services. The routes are optimal or not makes great influences to the service cost, working efficiency and customer satisfaction. The resolution of the Vehicle Routing Problem is beneficial for both the company and society. On one hand, it helps to improve the efficiency of delivery or service of the company, make full use of its vehicle resources and increase economic efficiency. More importantly, reasonable routing plan ensures the arrival to customers at the appointed time which will bring better service to the customers. On the other hand, for the society, rational routing planning can save vehicle resources, alleviate traffic congestion and reduce environmental pollution.

With the development of economic, more and more enterprises need to provide low-cost and high-level services. Logistics and other transportation industries are also increasingly inclined to short-distance and short-term transport. Therefore, the problem of vehicle routing problem with time windows (VRPTW) is very worthy of study.

A majority of the study's respondents indicated "cutting transportation cost" as a top challenge, followed by the "business process improvement" and "improved customer service". The enterprises are demanding greater innovation and technology advances while simultaneously remaining cost-conscious.

Coming to this dissertation, research is done regarding Vehicle Routing Problem with Time Windows (VRPTW), an NP-hard problem, meaning that it is almost impossible to obtain a perfectly optimized solution. Many algorithms till date are being proposed to optimize the solution which is why it is known as an NP-hard problem. This thesis considers two categories of the VRPTW problem: deterministic and robust problems. We study the impact of different optimization aspects on the problems solutions. For instance, we consider the significance of the thread parallelism in solving Vehicle Routing with Time Windows and how they can reduce the totality of the execution time. Moreover, we take the advantage of the clustering algorithms in order to solve the vehicle routing problem, and they have been proven to be highly effective and efficient.

1.2 Vehicle Routing Problem

Since the pioneer paper of Dantzig and Ramser, 1959 on the truck dispatching problem appeared at the end of the fifties of the last century, work in the field of the vehicle routing problem (VRP) has exploded dramatically. Using a method based on a linear programming formulation, the authors of this work produced by hand calculations a near-optimal solution with four routes of a fleet of gasoline delivery trucks between a bulk terminal and twelve service stations supplied by a terminal.

Nowadays, vehicle routing problem is considered as one of the most outstanding research achievement in the story of operations research and particularly in practice. There are important advances and new challenges that have been raised in the last few years due to technological innovations such as global positioning systems, radio frequency identification, and parallel computing (see for e.g Pillac et al., 2013, Montoya-Torres et al., 2015 and references therein).

The mathematical formulation of the problem can be represented on a graph $G = (N, A)$, where $N = \{o, 1, ..., n\}$ is the set of nodes, and $A = \{(i, j) : i, j \in A, i \neq j\}$ is the set of arcs. The node o represents the depot, and each other node is affected by a customer i. Each arc (i, j) is assigned to the travel cost c_{ij}, which, in general, is proportional to the travel time t_{ij} or the distance d_{ij} between i and j. The nominal service time is denoted by P_i^k for each vehicle k and node i within the time window $[a_i, b_i]$.

1.2.1 Some variants of VRP

After 60 years of its first definition, the vehicle routing problem (VRP) was and still is one of the most challenging fields of operations research because of its various practical applications and considerable hardness. Many variants have been created for the problem in order to take into consideration the real-world constraints. We first present a basic variation called in the literature Capacitated Vehicle Routing Problem (CVRP). The CVRP designs optimal delivery routes where each vehicle only travels one route, and there is only one central depot. Thus, The main purpose is to find a set of least cost routes, beginning and ending at the depot, that together cover a set of customers, besides no vehicle is loaded more than its capacity allows it to. In other words, a solution of CVRP is a set of routes which all begin and end in the depot, and which satisfies the capacity limitation, and the constraint that all the customers are served only once. The transportation cost can be improved by reducing the total traveled distance and by reducing the number of required vehicles (see for e.g 1.1).

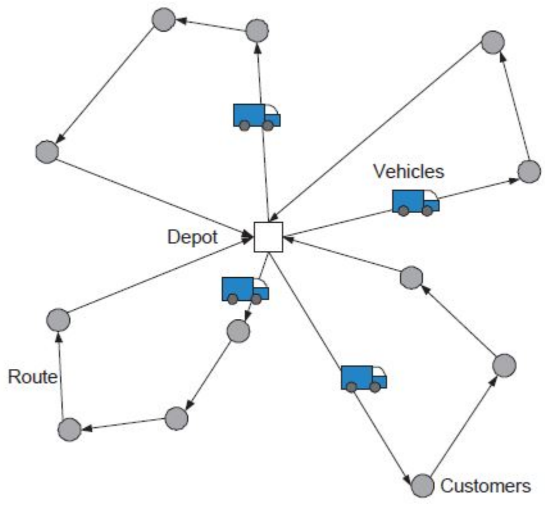

FIGURE 1.1: Vehicle Routing Problem.

This classical VRP has been extended in many ways by introducing additional real-life aspects or characteristics, resulting in a large number of variants of the VRP. One of the most important variants are those which deal with extra temporal constraints, the VRP with Time Windows (VRPTW), assumes that deliveries to a given customer must occur in a certain time interval, which varies from customer to customer (see for e.g 1.2).

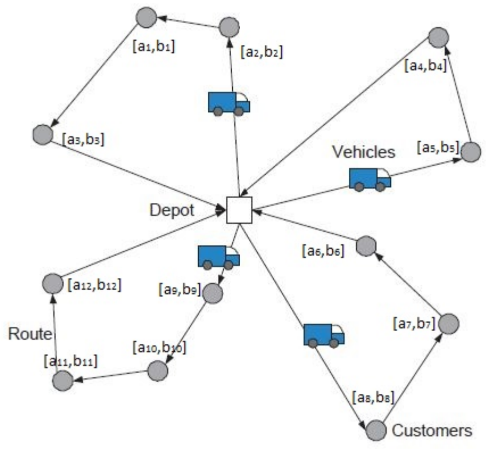

FIGURE 1.2: Vehicle Routing Problem with Time Windows.

VRP can be modeled with non identical vehicles. The typical variability that disturbs the homogeneity is the capacity of the vehicles, but there can be other factors such as different travel times, different costs or time windows for the vehicles. In the non identical (or multiple vehicle type) VRP, the vehicles can vary or there may exist categories of vehicles where an upper limit on the number of vehicles in each category is given in most cases.

In the VRP with Pickup and Delivery (VRPPD), we are concerned with the distribution of the goods, but we can also pick up goods from customers. The pick-up and drop-off must be done by the same vehicle, which is why the pick-up location and drop-off location must be included in the same route (Tasan and Gen, 2012). A related problem is the VRP with backhauls (VRPB), where a vehicle does deliveries as well as pick-ups in one route (Pradenas, Oportus, and Parada, 2013). Some customers require deliveries (referred to as linehauls) and others require pick-ups (referred to as backhauls). The combination of linehauls and backhauls has been proven very valuable to the industry.

A related variant called in the literature VRP with Multiple Compartments, when the vehicles transport several commodities, which must remain separate during the transportation, multiple compartment vehicles are used. The multiple compartment case has no influence on the main problem structure, but capacity constraints should be revised. If vehicle has m compartments, the capacity constraints must be modeled by m states instead of one.

In the same context, the Multi-Depot VRP (MDVRP) assumes that multiple depots are geographically spread among the customers (Montoya-Torres et al., 2015). In this situation, each depot can have its own fleet of vehicles or the vehicles may be based on different depots. For the first case, it is usually assumed that the vehicles must return to the same depot. For the second case, there can be a constraint such as the number of vehicles that arrive at a depot must be equal to the number of vehicles that leave the depot.

Another variant called Periodic VRP (PVRP) is used when planning is made over a certain period and deliveries to the customer can be made in different days (Hemmelmayer, 2014). For the PVRP, customers can be visited more than once, though often with limited frequency.

Vehicle routing problems in many industrial applications must take into account uncertain demand, traffic conditions, and/or service times (see for e. g. 1.3). The uncertainty present in a vehicle routing problem (VRP) can make a solution exhibit clear inefficiencies in specific outcomes of the uncertainty. For example, a vehicle may be directed to visit a customer that does not show. Different approaches have been proposed in order to handle uncertain events in a VRP, in demand, displacement time, and service time (Nasri et al., 2020a, Nasri, Hafidi, and Metrane, 2020b).

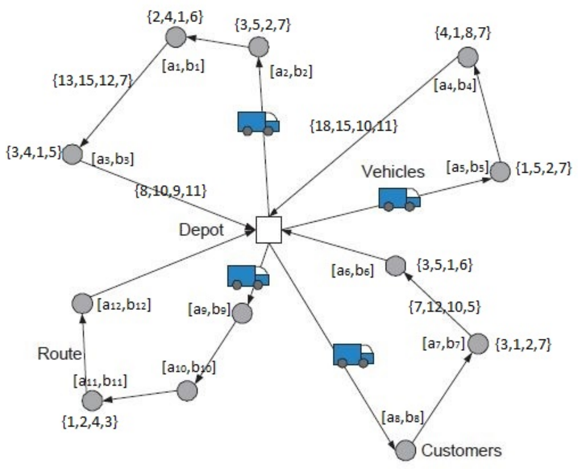

FIGURE 1.3: Vehicle Routing Problem with Time Windows under uncertain service and travel times.

1.2.2 Solution Methods for the deterministic case

For instance, three types of solution approaches can be used to solve these types of problems. First, the exact methods assert that the optimal is found if the method is given sufficiently in time and space. We cannot expect to construct exact algorithms which solve NP-hard problems. Second, the heuristics are solution methods that can quickly achieve a feasible solution in a reasonable quality. A special class called metaheuristics provides a high solution quality (see for e.g. Labadie, Prins, and Prodhon, 2016). This third class of solutions is also a special class of heuristic which provide a near-optimal solution and an error guarantee.

Exact methods

There have been many papers proposing exact algorithms for the VRPTW. The first of these papers was published by Kolen, Rinnooy, and Trienekens, 1987. Since then, many people have used exact algorithms for finding an optimal solution for the VRPTW. These exact algorithms can be classified in three groups:

- Dynamic Programming

- Lagrangean Relaxation based Methods
- Column Generation

Dynamic Programming

The first paper on dynamic programming for the VRPTW is the publication of Kolen, Rinnooy, and Trienekens, 1987. It is inspired from Christofides, Mingozzi, and Toth, 1981 which used Dynamic Programming for the VRP for the first time. The algorithm of Kolen, Rinnooy, and Trienekens, 1987 uses branch and bound approach in order to retrieve optimal solutions.

Branching is done by selecting a customer that is not forbidden and that does not appear in any route. At each branch and bound node, Dynamic Programming is used to calculate a lower bound on all feasible solutions. The authors solved problems up to 15 customers by this method in their paper.

Lagrangean Relaxation based Methods

There exist many papers that use Lagrangean Relaxation based Methods using different approaches. Variable splitting followed by Lagrangean Relaxation was used by (Jornsten, Madsen, and Sorensen, 1986, Madsen, 1988, Halse, 1992, and Fisher, Jornsten, and Madsen, 1997). Fisher, Jornsten, and Madsen, 1997 used K tree approach followed by Lagrangean Relaxation. Finally, Kohl and Madsen, 1997 applied shortest path with side constraints approach followed by Lagrangean Relaxation.

Variable splitting (or cost splitting) was first presented in the technical report of Jornsten, Madsen, and Sorensen, 1986, but no computational results were given in the paper. Madsen, 1988 used four different splitting approaches but they are not tested either. Halse, 1992 offered three approaches and he had implemented and tested one of these approaches.

Fisher, Jornsten, and Madsen, 1997 presents an optimal algorithm where the problem is formulated as a K tree problem with degree $2K$ on the depot. The VRPTW can be formulated as finding a K tree with degree $2K$ on the depot, degree 2 on the customers and subject to capacity and time constraints. This representation becomes equal to K routes. This algorithm was able to solve many of the clustered Solomon test problems but it could not solve any of the random given test problems.

Column Generation

Column generation has turned out to be an efficient method for a range of vehicle routing and scheduling problems. This approach is implemented previously by Desrochers, Desroiers, and Solomon, 1992 and Kohl, 1995. Column generation is based on the idea of initializing the linear program with a small subset of variables (by setting all other variables to 0) and computes a solution of this reduced linear program. Column generation used together with branch and bound is denoted as branch and price.

Desrochers, Desroiers, and Solomon, 1992 add feasible columns as needed by solving a shortest path problem with time windows and capacity constraints using dynamic programming. The LP solution obtained provides a lower bound that is used in a branch and bound algorithm to solve the integer set partitioning formulation.

By column generation, problems up to 25 customers can be solved optimally, but only few of the problems with 50 and 100 customers can be solved.

Heuristic methods

Since the VRPTW is proven to be NP-hard, non-exact algorithms are very popular for finding solutions. There are many papers that propose heuristic algorithms to the VRPTW.

Heuristic algorithms that build a set of routes are known as construction algorithms and are used to build an initial feasible solution for the problem. Usually, they are used to generate good feasible solutions. These algorithms can be classified as sequential and parallel algorithms. On the one hand, in a sequential algorithm one route is constructed initially and others are constructed when necessary. The first sequential construction (route building) algorithm was proposed by Baker and Schaffer, 1986. This algorithm can be interpreted as an extension of the Savings Heuristic of Clarke and Wright, 1964. The authors developed a sequential algorithm by defining the savings as a combination of distance and time feasibility. Then, Solomon, 1987 used a similar heuristic where the time feasibility aspect is not included in the savings function. The arcs that can be used are limited by the magnitude of the waiting times. It is required to check the violation of time windows when two routes are integrated. Reasonable results were reported for Savings Heuristic and the adopted versions. One of the sequential route building algorithms proposed by

Solomon, 1987 is Time Oriented Sweep Heuristic. It has two phases: a clustering phase which assigns customers to different clusters and a scheduling phase which solves a TSPTW problem by using the TSPTW heuristics proposed by Savelsbergh, 1985. On the other hand, in a parallel algorithm, many routes are constructed simultaneously. Potvin and Rousseau, 1993 proposed a parallelization of the Insertion Heuristics by creating many routes simultaneously. For the initialization of each route, the customer that is farthest from the depot is selected as a "center customer". Then, the customers are inserted to the best feasible insertion place. Russell, 1995 also adapted parallel insertion approach. Foisy and Potvin, 1993 also presented a parallel algorithm which is an Insertion Heuristic building routes simultaneously using the Solomon's heuristic to generate the initial center customers.

Another category of heuristics which is known as improving Heuristics, is based on the concept of neighborhood. Neighborhood concept has been used for at least 40 years for combinatorial optimization problems. One of the earliest references is Croes, 1958 which used the idea to improve the solutions of the Traveling Salesman Problem. Checking some or all the solutions in a neighborhood might reveal better solutions. This idea can be repeated starting at the better solution. At some point, no better solution can be found and a local optimum has been reached. This algorithm is called local search. Most of the improvement algorithms for the VRPTW use an exchange neighborhood to obtain a better solution. Two classical algorithms which were originally proposed for Traveling Salesman Problem are k-Opt and Or-Opt heuristics. The k-opt heuristic replaces a set of links in the route by another set of k links. The complexity of the heuristic is mostly affected by the size of k. For larger k values, the heuristic tends to give better results, but the computational time increases (see for e. g. Lin and Kernighan, 1981). The Or-Opt exchange, which was originally proposed by Or, 1976 for traveling Salesman Problem, removes a chain of at most three consecutive customers from the route and tries to insert this chain at all feasible locations in the routes. This heuristic is slightly modified by allowing the chain to be inserted in the same route and other routes.

Metaheuristic methods

In order to escape local optima and enlarge the search space, metaheuristic algorithms such as tabu search (TS), simulated annealing (SA), and genetic algorithm (GA) have been used to solve VRPTW.

Tabu search is based on the neighborhood search with local optima avoidance. The idea that lies under TS is systematically violating the feasibility conditions. At each iteration, the neighborhood of the current solution is explored and the best solution is selected as the new current solution. However, as opposed to a classical local search technique, the procedure does not stop at the first local optimum when no more improvement is possible. The best solution in the neighborhood is selected as the current solution even if it is worse than the current solution. The TS metaheuristic was the subject of series of papers (see for e. g. Potvin et al., 1996, Taillard et al., 1997, Bräysy and Gendreau, 2002).

In the same context, The simulated annealing was introduced by inspiring the annealing procedure of the metal working. Annealing procedure defines the optimal molecular arrangements of metal particles where the potential energy of the mass is minimized and refers cooling the metals gradually after subjected to high heat. In general manner, SA algorithm adopts an iterative movement according to the variable temperature parameter which imitates the annealing transaction of the metals (see for e. g. Chiang and Russell, 1996, Afifi, Dang, and Moukrim, 2013).

The genetic algorithm (GA) is a randomized search technique operating on a population of individuals, which form the solutions. A fitness value is calculated for each individual and the search is guided by this value. The GA generates solutions using techniques which are inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The GA outperforms all known metaheuristics that solves large-scale instances with high solution quality (see for e. g. Thangiah, 1995, Gehring and Homberger, 1999).

1.2.3 Solution approaches for the problem with uncertainties

Different approaches have been proposed to deal with uncertainties in a VRP problem either in demand, travel time and/or service time. Among them, the stochastic approaches of vehicle routing problem SVRP have been treated in series of papers (Dror, Laporte, and Louveaux, 1993, Dror and Trudeau, 1986, Gendreau, Laporte, and Séguin, 1996). The aim of this SVRP methodology is to find a near-best solution of the objective function responding to all possible data uncertainty. An alternative approach to handle the uncertain parameters is the robust optimization in which one can optimize against the worst scenario that can be generated from the source of uncertainty by using bi-objective function (Yousefi et al., 2017) and is immunized against this uncertainty I. Sungur and Dessouky, 2008. In this context, the literature coats a large number of applications such as scheduling (Goren and Sabuncuoglu, 2008, Hazir, Haouari, and Erel, 2010), facility location (Minoux, 2010, Baron, Milner, and Naseraldin, 2011, Alumur, Nickel, and Gama, 2012, Gülpinar, Pachamanova, and Canakoglu, 2013), inventory (Bienstock and Özbay, 2008, Ben-Tal, Boaz, and Shimrit, 2009), finance (Fabozzi et al., 2007, Gülpinar and Rustem, 2007, C. Pinar, 2007). In particular, the authors proposed a mathematical model for the robust optimization with uncertain demands (Moghaddam, Ruiz, and Sadjadi, 2012) and heterogeneous fleet (Noorizadegan, Galli, and Chen, 2012) and routing with capacity (I. Sungur and Dessouky, 2008, Gounaris, Wiesemann, and Floudas, 2013). For instance, this is equivalent to the deterministic case studied by Miller-Tucker-Zemlin formulation of the used VRP. We refer the reader to an excellent survey and tutorial of the robust vehicle routing proposed by . We note that uncertainty in travel cost could be handled using the robust combinatorial optimization approach, Wu, Hifi, and Bederina, 2017 have proposed a linear model evaluated on a set of random instances for the vehicle routing problem with uncertain travel time to improve the robustness of the solution, which enhance its quality compared to the worst case on a majority of scenarios. In the same spirit, Toklu, Montemanni, and Gambardella, 2013 have been treated the VRP problem with capacity and uncertain travel costs based on a variant of the ant colony algorithm to generate sets of solutions of uncertainty levels and to analyze their effects on the problem.

The stochastic approach is also applicable for the vehicle routing problem with time window constraints (VRPTW). Errico et al., 2016 have formulated the VRPTW with stochastic service times as a set partitioning problem and solve it by exact branch-cut-and-price algorithms. Precisely, they developed efficient labeling algorithms by choosing label components, determining extension functions, and developing lower and upper bounds on partial route reduced the cost to be used in the column generation step. Unlike this approach, robust optimization seeks to get good solutions for the VRPTW problem by only considering nominal values and deviations possible uncertain data. Many works tackled the vehicle routing problem with time windows and uncertain travel times (I. Sungur and Dessouky, 2008). Agra et al., 2012 presented a general approach to the robust VRPTW problem with uncertain travel times. Travel times belong to a demand uncertainty polytope, which makes the problem more complex to solve than its deterministic equivalent. The advantage of the addition in complexity is that the model from Agra et al., 2012 is more supple than the one from I. Sungur and Dessouky, 2008 and leads to less conservative robust solutions. Toklu, Montemanni, and Gambardella, 2013 have adapted their approach developed to solve the problem of VRPTW with uncertain travel times, whose objective is to minimize window time violation penalties by providing the decision makers a group of solutions found over several degrees of uncertainties considered. Agra et al., 2013 studied the VRPTW with uncertain travel times and proposed two robust formulations of the problem. The first extends the formulation of inequalities of resources and the second generalizes the formulation of inequalities of way. Their results show that the solution times are similar for both formulations while being significantly faster than the solutions times of a layered formulation recently proposed for the problem.

1.3 Presentation of the studied problem

1.3.1 Deterministic problem

We consider for the Vehicle Routing Problem (VRP), the following objective function:

$$\textbf{Min} \sum_{k \in V} \sum_{(i,j) \in A} d_{ij}\, x_{ij}^k$$

where the cost function $c(x) = \sum_{k \in V} \sum_{(i,j) \in A} d_{ij}\, x_{ij}^k$ refers to the total traveled distance d_{ij} and x_{ij}^k are binary decision variables that take the value 1 if vehicle k which belongs to the set of vehicles V uses the edge (i,j) and zero otherwise. The set of arcs of different routes is denoted as A and contains pairs of nodes (i,j). The constrained optimization problem must be accomplished under some constraints:

$$\sum_{k \in V} \sum_{j \in N} x_{ij}^k = 1 \qquad \forall\, i \in N, \tag{1.1}$$

This first constraint means that each customer must be visited once.

$$\sum_{j \in N} x_{0j}^k = 1 \qquad \forall\, k \in V, \tag{1.2}$$

The constraint (1.2) ensures that each tour begins from the depot.

$$\sum_{i \in N} x_{ih}^k - \sum_{j \in N} x_{hj}^k = 0 \qquad \forall\, h \in N \quad \forall\, k \in V, \tag{1.3}$$

The constraint (1.3) is a flow conservation, and consequently ensures that a vehicle arrives must leave from each node.

$$\sum_{i \in N} x_{i0}^k = 1 \qquad \forall\, k \in V, \tag{1.4}$$

This fourth constraint guarantees that each tour returns to the depot.

Since the service time P_i^k at any client i by vehicle k begins inside a given time interval $[a_i, b_i]$, we require an additional constraint.

$$a_i \le P_i^k \le b_i \qquad \forall\, i \in N \quad \forall\, k \in V, \tag{1.5}$$

The time windows considered here is hard, i.e. they cannot be violated, if the vehicle arrives earlier than required at a client i, it must hold up until the time window $[a_i, b_i]$

opens and moreover it is not permitted to arrive late which leads:

$$P_i^k + d_{ij} - P_j^k \leq M(1 - x_{ij}^k) \qquad \forall\, i \in N \quad \forall\, j \in N \setminus \{0\} \quad \forall\, k \in V. \tag{1.6}$$

with M a great value. For instance, a review how to handle hard and soft time window constraint in VRPTW problem can be found in Hashimoto et al., 2010.

1.3.2 Robust problem

To model the uncertainty in travel times and service times in the presence of time windows, a step-wise (layered) formulation is used. Based on the approach of Bertsimas and Sim, 2003, we assume that the travel times and service times are uncertain, and that they take their values respectively in the intervals $[t_{ij}, t_{ij} + \Delta_{ij}]$ and $[P_i, P_i + \delta_i]$. where t_{ij} and P_i are the nominal values, Δ_{ij} and δ_i represent the maximum deviations. We also define the sets of uncertainties associated with these times by:

$$U_t = \{\tilde{t} \in \mathbb{R}^{|A|} \;/\; \tilde{t}_{ij} = t_{ij} + \Delta_{ij}\epsilon_{ij}, \sum_{(i,j)\in A} \epsilon_{ij} \leq \Gamma, 0 \leq \epsilon_{ij} \leq 1, \; \forall (i,j) \in A\}$$

and

$$U_P = \{\tilde{P} \in \mathbb{R}^{|N|} \;/\; \tilde{P}_i = P_i + \delta_i\omega_i, \sum_{i\in N} \omega_i \leq \Lambda, 0 \leq \omega_i \leq 1, \; \forall i \in N\}$$

Γ and Λ are two degrees of uncertainty defined to control the number of travel times and service times uncertain. They vary respectively between 0 and $\mid N \mid + \mid V \mid$, and 0 and $\mid N \mid$. Thus, when $\Gamma = 0$ and $\Lambda = 0$ the robust case coincides with the deterministic case, and when $\Gamma = \mid N \mid + \mid V \mid$ and $\Lambda = \mid N \mid$ is the worst case where all travel times and service times are supposedly uncertain and simultaneously reach their worst values.

Robust optimization seeks to obtain good solutions for all the possible realizations of the uncertainties without needing to define the laws of probability and considering only the nominal values and the possible deviations of the uncertain data. We introduce uncertainties by replacing the objective function by:

$$\textbf{Min} \; (\sum_{k\in V} \sum_{(i,j)\in A} x_{ij}^k t_{ij} + \max_{\{\Psi / \Psi \subset A, |\Psi| = \Gamma\}} \sum_{k\in V} \sum_{(i,j)\in \Psi} x_{ij}^k \Delta_{ij})$$

And the constraint 1.6 treating the time windows by this:

$$P_i^k + t_{ij} + \delta_i \nu_i^\theta + \Delta_{ij} \mu_{ij}^\Psi - P_j^k \leq (1 - x_{ij}^k) M$$

$$\forall (i \in N),\ \forall (j \in N \setminus \{0\}),\ \forall (k \in V),\ \forall (\theta \subset N) \mid \theta \mid = \Lambda,\ \forall (\Psi \subset A) \mid \Psi \mid = \Gamma$$

where ν_i^θ and μ_{ij}^Ψ are two indicator functions. ν_i^θ takes the value 1 when $i \in \theta$ and μ_{ij}^Ψ takes 1 when $(i, j) \in \Psi$

1.4 Contribution of the Dissertation

This dissertation considers the optimization of real world field vehicle routing problem with time windows. The aim is to find a set of least cost routes, minimizing a travel distance of vehicles, starting and ending from a depot, to serve some customers. We developed our research by investigating two categories of this problem: deterministic and robust problems. The main contributions of this dissertation are summarized as follows:

- We suggest a modified adaptive large neighborhood search (MALNS) to tackle the vehicle routing problem with time windows (VRPTW). In this context, we integrate the modified choice function as selection mechanism in the ALNS method which appears well-suited for the VRPTW. The competitiveness of our proposed method is demonstrated on the Solomon's benchmark instance of VRPTW. We conclude that our MALNS algorithm outperforms the classical ALNS in terms of solution quality.
- We propose a new approach which fits into the class of algorithms Cluster first - Route second to deal with the VRPTW problem. The suggested strategy consists of two phases, clustering and routing. The first one aims to define a set of cost-effective feasible clusters using k-medoid algorithm within an effective spatio-temporal distance similarity. While the second phase aims to select the adequate routes, considering that each cluster can be considered as specific VRPTW sub-problem.
- We introduce a new multi-threading parallel adaptive large neighborhood search to solve the well known optimization VRPTW problem. More precisely, we applied a parallel computing to the greedy insertion involved in the initial block and the destroy/repair operators incorporated into the ALNS process. Our algorithm is shown to be able to obtain a good solution in a reasonable time without too much compromise on the quality of the solution. In order to ensure the quality commitment of the solution, we integrate the K-means clustering algorithm as a pre-processing treatment.
- The robust vehicle routing problem with time windows under both travel times and service times uncertainties is considered. For this purpose, a new resolution robust approach has been suggested to minimize the total distance

of the travel time in the presence of the maximum deviations of possible uncertain data. In this contrast, we generated all possible scenarios by using Metropolis-Monte Carlo simulation and we opt for the adaptive large neighborhood search ALNS algorithm to solve each sub-problem related to each scenario. In order to study the feasibility of the resulting solution, efficient mechanisms have been developed; the first concerns the verification of the robustness, while the second takes into consideration the evaluation of the solution on the worst case.

- As far as the adopted approach derives the best robust solution that responds to all uncertainties, it still suffer from lengthy computational times; partly because the generation of scenarios, as well as the ALNS block are time-consuming. As an alternative to remedy this problem, we introduce a procedure for the thread parallelism in the Monte Carlo block, and we use the parallel ALNS proposed by Røpke, 2009. This leads to four different robust approaches combining the (sequential/parallel) Monte Carlo algorithm and the (sequential/parallel) ALNS. The considered approaches are tested on Solomon's benchmark instance of VRPTW and lager instances generated randomly. Accordingly, we can offer a decision-making solution that provides great protection against delays in a reasonable running time.

1.5 Organization of the Dissertation

After presenting the field of Study, the thesis is divided into two parts. The first part covers the deterministic Vehicle Routing Problem with Time Windows (VRPTW). It concerns the most popular variant of VRP where each customer has a time window constraint, it became among the most studied variants of routing problems due to its wide range of applications. This part contains three sections: 2.2, 2.3 and 2.4.

After presenting the problem, reviewing the literature and giving its mathematical formulation, Section 2.2 exposes the Adaptive Large Neighborhood Search metaheuristic introduced by Pisinger and Ropke, 2007, including the presentation of different destruction and insertion heuristics involved therein. Then, in section 2.3, we suggest two different alternatives to enhance the ALNS process. The first approach entitled a modified adaptive large neighborhood search (MALNS), consists on a framework of metaheuristic designed to solve the vehicle routing problem with time windows. We integrate a new mechanism to deal with the Adaptive Large Neighborhood Search (ALNS) algorithm applied to the problem. The main challenge is to highlight intensification over diversification within the heuristic search process. The second approach is a hierarchical approach composed of two stages as "Cluster first - Route second". In the first stage, customers are assigned to vehicles using K-medoid clustering algorithm within a spatio-temporal similarity distance. In the second stage, the VRPTW is solved using three distinct routing algorithms (i.e., ALNS, GA and VNS). In section 2.4, we provide a new approach of parallel adaptive large neighborhood search algorithm designed to solve the vehicle routing problem with time windows. The main challenge is to obtain a reduced real-time solution without compromising the quality of the solution specially for large instances.

The second part deals with a relatively new variant of VRPTW which extends from the last one and studies the problem under uncertain travel and service times. The problem is therefore called the robust VRPTW (RVRPTW). After introducing the problem and its state of art, section 3.4 presents a new resolution approach for the robust problem based on the implementation of an adaptive large neighborhood search algorithm and the use of efficient mechanisms to derive the best robust solution that responds to all uncertainties with reduced running times. Then, section 3.5, study the effect of multi-threading parallelization of this resolution approach blocks on the running time and the objective function. The contribution to the previous section lies first on the choice of the efficient Parallel Adaptive Large Neighborhood Search (PALNS) metaheuristic of Røpke, 2009, which leads to a reduced running time. Moreover, we used our parallel version of the Metropolis Monte Carlo algorithm to generate all possible realizations and to transform the problem under uncertainties to a set of deterministic sub-problems. Based on the efficient implementation of Røpke, 2009, different combinations (sequential/parallel) of the Monte Carlo algorithm and ALNS are then performed.

At the end, we finish with a general conclusion including a synthesis of the contributions presented in this dissertation and some perspectives and future works.

Chapter 2

Deterministic Vehicle Routing Problem with Time Windows

2.1 State of the Art

Since the publication of the seminal work of Dantzig and Ramser, 1959, the Vehicle Routing Problem (VRP) has been the subject of a great deal of research. The VRP has extended its scope by introducing additional real-world facets to derive a large number of variants different from the one introduced by Dantzig and generalized by Clarke and Wright, 1964. For given customers with known locations and demands, the goal of the VRP problem in its general form is to design a set of minimum cost routes that serve a number of geographically distributed customers around the central depot, and fulfilling specific constraints of the problem.

The VRP belongs to the class of operations research and combinatorial optimization problems. Exact method fails to solve this problem of optimization for large instances more than one hundred customers except the Traveling Salesman Problem (TSP) where instances with several thousand nodes can be solved optimally on a regular basis, see for instance Gutin and Punnen, 2002 and Toth and Vigo, 2002. The VRP is an NP-hard problem, see for e.g Lenstra and Kan, 1981. It is hard to solve problem with small instances optimally above 50 customers as reported by Azi, Gendreau, and Potvin, 2010.

For a comprehensive state of the art classification and review of the VRP, we refer the reader to the paper of Eksioglu, Vural, and Reisman, 2009 and to an adapted version of the taxonomy suggested by Braekers, Ramaekers, and Nieuwenhuyse, 2015. Another recent concise review of existing VRP problem features and applications can be found in the paper by Vidal, Laporte, and Matl, 2019.

The famous extension of the VRP is the VRP with Time Windows (VRPTW) (Bräysy and Gendreau, 2005a, Kallehauge et al., 2005 and Ticha et al., 2017), where the service at any customer starts within a certain time interval called a time window. Time windows are defined as hard (or strict) when it must be satisfied, while it is called soft if it can be violated and allow deliveries outside the boundaries against a penalty added to the objective function.

Combinatorial optimization problems of this kind are non-polynomial-hard (NP-hard) and are hence best solved by using heuristics and metaheuristics which are often more suitable for real-life problems which are large scale in nature. Their complexity may emanate from various sources, such as customers, vehicles, shipments and physical infrastructure, all interacting in different ways. The most important metaheuristics used to solve the VRPTW are Tabu Search (TS) (Potvin et al., 1996, Taillard et al., 1997, Bräysy and Gendreau, 2002), the Simulated Annealing (SA) (Chiang and Russell, 1996, Afifi, Dang, and Moukrim, 2013), the Adaptive Large Neighborhood Search (ALNS) (Røpke and Pisinger, 2006), Genetic Algorithm (GA) (Thangiah, 1995, Gehring and Homberger, 1999), Ant Colony Optimisation Algorithm (ACO) (Gambardella, Taillard, and Agazzi, 1999, Tan, Zhuo, and Zhang, 2006) and Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997, Dhahri et al., 2016). As survey of metaheuristics for VRPTW can be found in Bräysy and Gendreau, 2005a.

The survey of Kallehauge et al., 2005 is given in the context of column generation in general and the focus is therefore the path formulation of the VRPTW. These surveys also give a status on the computational success of the state of the art algorithms proposed in the literature. An other interesting way to treat such problem is to combine metaheuristics, mathematical programming, constraint programming and machine learning, see for e.g. Talbi, 2013.

Over time, researchers have developed more powerful algorithms that find improved solutions to both real-world VRPTW instances and well studied Solomon, 1987 and Gehring and Homberger, 1999 benchmark instances by building on previous work and by taking advantage of faster computational resources.

For real problems, the instances are massive scale and the input data changes in an unexpected way, the major challenge is to optimize the execution time of the algorithm. Therefore, the need of the parallelization has become an indispensable and even inevitable technique to tackle this problem. But, it is important to keep in mind that the relevant noun counterpart which is the quality of the solution can be affected. For instance, we try to select a "good" solution in a reasonable time without too much compromising the quality of the solution. An efficient method to find a compromise between the time of execution and the quality of a solution will be studied in this work.

Following these ideas, Røpke, 2009 has proposed parallel ALNS applied to the Travel Salesman Problem with Pickup and Delivery, and the Capacitated Vehicle Routing Problem such that each worker thread obtains a copy of the current solution and performs destroy and repair operations on its local copy in order to generate a local solution which will be compared to the global and best found solutions. In the same spirit, Hemmelmayer, 2014 has proposed a parallel variant of the Large Neighborhood Search LNS to solve the Multi-depot and Periodic Vehicle Routing Problem. On the same topic, Pillac et al., 2013 has presented a parallel version of the ALNS by adding a set covering post-optimization model that combines the tours generated throughout the search to assemble a better solution.

In this chapter, we devote the next section to the exposition of the Adaptive Large Neighborhood Search metaheuristic introduced by Pisinger and Ropke, 2007, including the presentation of different destruction and insertion heuristics involved therein. Within the section 2.3, we put forward two different alternatives to enhance the ALNS process. The first approach called modified adaptive large neighborhood search (MALNS), consists on incorporating the modified choice function proposed by Drake, Özcan, and Burke, 2012, into the selection phase of the ALNS in order to efficiently manage the destroy and repair operators. The second approach fits into the class of algorithms cluster first - route second. Indeed, changing the way of clustering may change the efficiency of the solution of the Vehicle Routing Problem with Time Windows. This fact, can be demonstrated by inserting a pre-processing step based on the K-medoid algorithm. In this stage, we subdivide the group of nodes of the general problem into small sets of customers which represent sub-problems. Then, the detail of the description of the ALNS multi-threading parallel approach is the core the section 2.4. Different operators involved in destruction and repair processing are discussed. Finally, we conclude this chapter and outline directions for future research.

2.2 Adaptive Large Neighborhood Search (ALNS)

The ALNS is a metaheuristic proposed by Røpke and Pisinger, 2006 as an extension of the Large Neighborhood Search (LNS) heuristic presented by Shaw, 1998. It is an adaptive approach employed to ameliorate an incumbent solution. The basic idea of this algorithm is to improve a given initial solution by exploring many large neighborhoods. This can be done by the application of various destroy and repair operators. More precisely, the destroy operator removes nodes from the solution. We obtain then an incomplete or infeasible solution d(x). The repair operator reinserts the removed nodes at more favored position which leads to a new feasible (complete) solution r(d(x)). The resulting solution will be accepted or rejected based on a Hill Climbing acceptance criteria which only accepts solutions that are better than the current one. This process will be terminated when a stop criterion is met.

It is worth mentioning, that ALNS uses an adaptive layer in the step where the operators are selected, by using a score ϕ_j which measures the best performance of an operator j during the search to choose the most successful one. The adaptiveness lies in a roulette wheel mechansim which selects an operator j with a probability $\frac{\phi_j}{\sum_i \phi_i}$. During M iterations, the score ϕ_j is updated and the probabilities of selecting an operator are recalculated.

In order to design an ALNS algorithm for a given optimization problem we need to:

- Choose a number of fast construction operators which are able to construct a full solution.
- Select a number of destruction operators. It might be sufficiently important to choose the destruction operators that are expected to work well with the construction operator, but it is unnecessary.

Here is the detailed algorithm:

Initial solution generation

In order to deal with the initial solution, we apply a greedy algorithm, which will be used in the reconstruction phase of the ALNS. This operator aims to insert the non-inserted nodes by testing the different possible configurations and then giving a feasible solution. It is not necessary that the completion time of the initial solution be minimal, as this solution will be further enhanced using the ALNS method.

Algorithm 1 adaptive large neighborhood search

```
Construct a feasible solution $x$ ; set $x^b = x$
repeat
    Choose a destroy neighborhood $d$ and a repair neighborhood $r$ using roulette
    wheel selection based on previously obtained scores $\pi_j$
    Generate a new solution $x^t$ from $x$ using the heuristics corresponding to the
    chosen destroy and repair neighborhoods
    if $x^t$ can be accepted then
        $x = x^t$
    end if
    if $c(x^t) < c(x)$ then
        $x^b = x^t$
    end if
    Update scores $\pi_j$ of $d$ and $r$
until Stop criteria is met
return $x^b$
```

Solution destruction

During the destruction phase, we put forward three different removal methods to maintain the diversity during the searching process and to define the neighborhood to explore at each iteration. Each removal method aims to remove a predefined number of nodes.

The first operator is known as proximity operator. Its objective is to select close clients according to a spatio-temporal measure Shaw, 1998, and then remove clients engendering the higher value of this measure.

Using the same technique, the route portion operator comes to give more flexibility than the proximity operator to change the routes. The principle consists in choosing a pivot client owned to a road and remove it as well as its adjacent clients. Then, we calculate the spatio-temporal measure, with the objective to select a second client belonging to another route but close to the initial pivot. This second pivot is removed from the solution as well as its adjacent clients and so on until all clients will be removed.

The third operator is referred to as longest detour operator. The interest of this operator is to remove the customers that lead to the largest cost increases for servicing them. For more details, we refer the reader to PrescottGagnon, Desaulniers, and Rousseau, 2009. The algorithms of the used destroy operators can be found in Appendix.

Solution reconstruction

Solomon's insertion heuristic (1987) presented a technique for choosing the new customer to be inserted into a route using two criteria $c_1(i,u,j)$ and $c_2(i,u,j)$ to select customer u for insertion between adjacent clients i and j in the current partial route. The primary criterion c_1 calculates the best feasible insertion place in the current route for each unrouted client u as

$$c_1(i(u),u,j(u)) \quad = \quad \min_{\rho=1,\ldots,m} c_1(i_{\rho-1},u,i_\rho) \tag{2.1}$$

The second criterion c_2 selects the new inserted customer:

$$c_2(i(u^*),u^*,j(u^*)) \quad = \quad \max_{u}\{c_2(i(u),u,j(u))|u \textit{ is unrouted and route is feasible}\} \tag{2.2}$$

Customer u^* is then inserted into the route between $i(u^*)$ and $j(u^*)$. The measurement of an insertion place c_1 depends on two factors: the increase in total distance of the current route after the insertion, and the delay of service start time of the customer following the new inserted customer. To be more precise, $c_1(i,u,j)$ is calculated as:

$$c_1(i,u,j) \quad = \quad \alpha_1\ (d_{iu}+d_{uj}-\mu\ d_{ij})+\alpha_2\ (b_{ju}-b_j), \tag{2.3}$$

$$\alpha_1+\alpha_2 \quad = \quad 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0, \quad \mu \geq 0, \tag{2.4}$$

where $d_{iu}+d_{uj}$ is the new distance between two nodes i and j after the insertion, b_j is the previous service start time, d_{ij} is the old distance between i and j and b_{ju} is the new service start time of customer j after the insertion of customer u. The criterion $c_2(i,u,j)$ is calculated as following

$$c_2(i,u,j) \quad = \quad \lambda\ d_{0u}-c_1(i,u,j), \lambda \geq 0 \tag{2.5}$$

where the parameter λ is used to define how much the best insertion place for an unrouted customer depends on its distance from the depot and extra time required to visit the customer by the current vehicle.

Roulette wheel selection

For each iteration of the destruction phase, a roulette-wheel procedure is applied to select a method for generating the neighborhood (nodes to be removed). During the search process, the ALNS maintains a score φ_j which measures the best performance of an heuristic j in the past iterations. The roulette wheel selection consists in selecting an heuristic j with a probability $\frac{\varphi_j}{\sum_i \varphi_i}$. During M iterations, the score φ_i is reset and the probabilities of choosing an heuristic are recalculated (Pisinger and Ropke, 2007).

2.3 Improvement of the Adaptive Large Neighborhood Search

In this section, we put forward two different alternatives to enhance the ALNS process. The first approach is the modified adaptive large neighborhood search (MALNS). It consists on a framework of metaheuristic designed to solve the vehicle routing problem with time windows. We integrate a new mechanism to deal with the Adaptive Large Neighborhood Search (ALNS) algorithm applied to the problem. The main challenge is to highlight intensification over diversification within the heuristic search process. In this context, we incorporate the process of the choice function proposed by Drake, Özcan, and Burke, 2012 into the ALNS algorithm. Therefore, several destroy/repair methods is combined to explore multiple neighborhoods within the same search and defined implicitly the large neighborhood. The second alternative is a hierarchical approach composed of two stages as "Cluster first - Route second". In the first stage, customers are assigned to vehicles using K-medoids clustering algorithm within a spatio-temporal similarity distance. In the second stage, the VRPTW is solved using three distinct routing algorithms (i.e., ALNS, GA and VNS).

2.3.1 The Choice Function at the service of the ALNS

In this part, we give a detailed exposition of our MALNS algorithm for solving the VRPTW. MALNS is a heuristic solution approach, which integrates the mechanism of the modified choice function (MCF) in the adaptive large neighborhood search (ALNS), in order to guide the research to areas where high-quality solutions are intended by seeking a trade-off between diversification and intensification.

The modified choice Function

The Modified Choice Function (MCF) is an efficient technique presented by Drake, Özcan, and Burke, 2012 as an extension of the original choice function of Cowling, Kendall, and Soubeiga, 2001. The idea behind this method is to dynamically control the selection of heuristics on the basis of a combination of three different measures. Thereby, the heuristic to be selected must have the higher score F_t.

The first measure f_1 reflects the past performance of each single heuristic. This measure is represented by the equation:

$$f_1(h_j) = \sum_n \phi^{n-1} \frac{I_n(h_j)}{T_n(h_j)}$$

where $I_n(h_j)$ presents the change in fitness function, $T_n(h_j)$ is the time it takes the heuristic h_j to produce a solution for an invocation n, and ϕ is a parameter from the interval $[0, 1]$ highlighting the recent performance.

The second measure f_2 tracks the dependency between a pair of heuristics (h_k, h_j), by considering their past performance when selected consecutively. The formula of this measure is given as follows:

$$f_2(h_j) = \sum_n \phi^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)}$$

where $I_n(h_k, h_j)$ presents the change in fitness function, $T_n(h_k, h_j)$ is the time it takes to call the heuristic h_j immediately after h_k for an invocation n.

The third measure f_3 notes the elapsed time $(\tau(h_j))$ since an heuristic h_j was last called. This gives the heuristics which are inactive for certain time, an opportunity to be selected.

$$f_3(h_j) = \tau(h_j)$$

The formulation of the modified choice function is given as follows:

$$F_t(h_j) = \phi_t f_1(h_j) + \phi_t f_2(h_k, h_j) + \delta_t f_3(h_j)$$

where t denotes the number of invocations of heuristic h_j indicating an improvement by the used heuristic.

The measures f_1 and f_2 bring intensification to the search process while the measure f_3 supports diversification by giving a chance to inactive heuristics to be selected. This is possible by the incorporation of the parameters ϕ_t and δ_t. Where ϕ_t is an intensification parameter which weights f_1 and f_2 respectively, and δ_t is the relative weight to f_3 and hence it is defined to control the diversification degree.
At each iteration, if the objective value improves, the value of ϕ_t is increased while δ_t is concurrently decreased. Conversely, ϕ_t is decreased and δ_t is increased when the objective value does not improve. The parameters ϕ_t and δ_t are expressed in the following way:

$$\phi_t(h_j) = \begin{cases} 0.99, & \textit{if the objective value improves} \\ max\{\phi_{t-1} - 0.01, 0.01\} & \textit{otherwise} \end{cases}$$

and

$$\delta_t(h_j) = 1 - \phi_t(h_j)$$

The Modified Adaptive Large Neighborhood Search (MALNS)

The idea behind the MALNS is to efficiently explore the search space using many large neighborhoods. The task consists on incorporating the modified choice function into the selection phase of the ALNS while sparing the whole process. In other words, the destroy and repair operators will not be selected by the roulette wheel mechanism as in the original ALNS, but they will be rather selected by the MCF. Here is the detailed algorithm of the MALNS method:

Algorithm 2 Modified Adaptive Large Neighborhood Search

```
Construct a feasible solution $x$ ; set $x^b = x$
repeat
    Choose a destroy neighborhood $d$ and a repair neighborhood $r$ using the modified choice function based on the obtained scores $F_t$
    Generate a new solution $x^t$ from $x$ using the heuristics corresponding to the chosen destroy and repair neighborhoods
    if $x^t$ can be accepted then
        $x = x^t$
    end if
    if $c(x^t) < c(x)$ then
        $x^b = x^t$
    end if
    Update scores $F_t$ of $d$ and $r$
until Stop criteria is met
return $x^b$
```

For the sake of completeness, we will describe the initialization step as well as the removal and insertion operators involved in the destroy/repair block to ensure the diversity during the searching process.

To deal with the initial solution, we used the greedy insertion heuristic introduced by Solomon, 1987. This method consists of finding the best location of a given node by testing the different possible configurations. More explicitly, the algorithm selects the best possible insertion place in the present route for each non inserted node under two considerations: the increase in total cost of the present route after the insertion, and the delay of service start time of the client following the new inserted client. This process ends when all deleted nodes will be inserted.

During the phase of destruction, we adopt three different removal heuristics. The first operator is referred to as proximity operator. This operator aims to delete a set of customers that are similar in terms of a spatio-temporal measure (Shaw, 1997 and Shaw, 1998). In the same manner, the route portion operator gives more flexibility of change on the routes by selecting a pivot client attributed to a route and remove it with its adjacent. The third operator is known as longest detour operator and tries to remove the customers that lead to the largest increase of the cost of the current solution. For more details about those operators, we refer the reader to PrescottGagnon, Desaulniers, and Rousseau, 2009 and references therein.

After the destruction phase, the obtained solution must be repaired in order to get a feasible solution. Therefore, We use two repair operators. First, the greedy insertion Solomon, 1987 tries to select the location that reduces the cost of insertion over all nodes and routes. Second, the regret insertion (Diana and Dessouky, 2004) defines a regret value which is the cost difference of inserting the customer i in its best route and its second best route. Thereby, customers with the highest value should be inserted first.

Computational experiments

In order to examine the performance of the proposed MALNS, we accomplished several computational tests. The algorithm was examined on a group of small instances in reference to the benchmark of Solomon, 1987, and its extension the instances of Gehring and Homberger, 1999:

- Set R contains problems with randomized customers.
- Set C contains problems with clustered customers.
- Set RC contains problems with both randomized and clustered customers.

As the examined algorithms are based on the related performance, the Modified Adaptive Large Neighborhood Search (MALNS) and the Adaptive Large Neighborhood Search (ALNS) are compared to the competition entries independently. Therefore, both of algorithms use three distinct destroy operators namely the proximity operator, the route portion operator and the longest detour operator. And two repair operators which are the greedy insertion and the regret insertion.

The algorithms were implemented in Java 7, compiled with Intel compiler Celeron 1.80 GHz core i5 with 8GB RAM. The MALNS approach was run for 15600 iterations and was applied 10 times to each instance. In this section, we will give the description of the impact of our method following different parameters:

Objective function

Table 2.1 illustrates the MALNS improvement results in terms of objective value for all instance groups compared to the ALNS algorithm. The first column defines the instance group, the second column contains the results of the ALNS, and the third column our MALNS algorithm results.

Instance	Initial solution	ALNS solution	MALNS solution
R101	1747.12	1650.80	1645.79
C101	929.21	828.94	828.94
RC101	1793.01	1708.80	1701.21
R201	1310.64	1253.23	1253.23
C201	682.52	591.56	591.56
RC201	1516.79	1406.94	1406.94
R121	4896.01	4819.12	4796.26
C121	2807.20	2704.57	2704.57
RC121	3725.37	3606.06	3606.06
R221	4603.41	4513.10	4483.16
C221	2063.76	1931.44	1931.44
RC221	3681.32	3605.40	3427.37
R141	10741.11	10639.75	10512.51
C141	7306.13	7152.06	7152.06
RC141	9652.33	9127.15	8724.36
R241	9961.12	9758.46	9594.32
C241	4837.71	4116.33	4116.33
RC241	7949.65	7471.01	7003.61
R161	23027.04	22838.65	22145.03
C161	14296.24	14095.64	14095.64
RC161	18215.52	17924.88	17432.87
R261	22210.38	21945.30	19806.15
C261	8344.17	7977.98	7977.98
RC261	14967.21	14817.72	14111.79
R181	39891.04	39315.92	38614.81
C181	25774.19	25184.38	25184.38
RC181	32626.95	32268.95	31316.12
R281	34260.80	33816.90	30641.10
C281	12212.17	11687.06	11687.06
RC281	23878.06	23289.40	22116.02
R1101	58014.95	56903.84	55243.64
C1101	43154.49	42488.66	42488.68
RC1101	49389.09	48702.83	47530.42
R2101	46157.93	45422.58	43994.26
C2101	17535.18	16879.24	16879.24
RC2101	35908.40	35073.70	33104.52

TABLE 2.1: Comparison of objective functions between the ALNS and MALNS

From the table, we can observe that the MALNS method outperforms the ALNS method, and yields better results in terms of solution quality, the gain average percent can reach 10.36%.

Execution time

The table 2.2 presents the execution time of our method compared to ALNS. We computed then the gain average percent (GAP) which is the absolute difference between execution time of the MALNS and the ALNS divided by the magnitude of the execution time of the ALNS.

Instance	ALNS	MALNS	gap (%)
R101	368	452	22.82
C101	325	396	21.84
RC101	351	430	22.50
R201	422	510	20.85
C201	365	437	19.72
RC201	415	500	20.48
R121	490	568	15.91
C121	435	504	15.86
RC121	454	525	15.63
R221	588	679	15.47
C221	505	581	15.04
RC221	530	613	15.66
R141	581	655	12.73
C141	560	627	11.96
RC141	501	564	12.57
R241	736	824	11.95
C241	703	787	11.94
RC241	718	801	11.55
R161	556	606	8.99
C161	680	741	8.97
RC161	619	672	8.56
R261	749	816	8.94
C261	817	890	8.93
RC261	788	858	8.88
R181	692	726	4.91
C181	708	743	4.94
RC181	697	731	4.87
R281	852	894	4.92
C281	937	983	4.90
RC281	917	962	4.90
R1101	884	901	1.92
C1101	910	928	1.97
RC1101	903	921	1.99
R2101	1318	1344	1.97
C2101	1361	1388	1.98
RC2101	1331	1357	1.95

TABLE 2.2: Comparison of runtime in seconds between the ALNS and MALNS

It is clear from the table above, that our MALNS takes more time of execution compared to the ALNS, with a percentage gap of an average between 22.84% and 1.95%. This can be explained by the fact that the Modified choice function adopts three different measures to choose the adequate operator to use, in the contrast of the roulette wheel selection which uses only the recent performance of the operator to favor the most appropriate one. When the instance size increases, the execution time of our approach converges to the ALNS one.

2.3.2 A Cluster first - Route second approach for solving the VRPTW

In this section, we propose a new approach which fits into the class of algorithms Cluster first - Route second to deal with the VRPTW problem. The strategy consists of two phases, clustering and routing. The first phase aims to define a set of cost-effective feasible clustering using k-medoid algorithm within an effective spatio-tamporal distance similarity which is totally appropriate to the nature of the VRPTW given that it considers both the spatial and temporal dimensions of the problem , while phase II is devoted to select the adequate routes, considering that each cluster corresponds to a specific VRPTW subproblem, we attempt to solve it by applying three different routing algorithms (i.e., Adaptive Large Neighborhood Search (ALNS), Variable Neighborhood Search (VNS) and Genetic Algorithm (GA)) separately in order to validate the results found in the first level. It is worth pointing that the choice of the K-Medoid was not arbitrary since it is more robust to noise and outliers and it is more flexible to be used with any similarity measure in contrast of other partitioning techniques which are not sensitive to noisy data or must be used only with distances that are consistent with the mean (e.g. K-Means). The used similarity measure within the K-Medoid algorithm was compared to some partitioning algorithms from the literature such as K-Means algorithm, and leads to solutions that have lower route costs no matter what metaheuristics are used for routing.

Spatio-temporal distance

In practice, it is interesting to pay attention to the dynamic characteristics of the problem. Thus, assigning two customers which have a close spatial distance while their time windows of service are far is inefficient, since the related counterpart which is the waiting time will be increased and thereby missing opportunities to serve other customers. Therefore, the spatio-temporal measure seeks to explore the spatial and temporal similarities between customers in terms of both the travel distance and time windows aspects.

The generalized equation of the spatio-temporal distance is proposed as follows:

$$ST_{ij} = \alpha_1 \, d_{ij} + \alpha_2 \, T_{ij}, \tag{2.6}$$

$$\alpha_1 + \alpha_2 = 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0 \tag{2.7}$$

where d_{ij} denotes the spatial distance between two customers i and j. The parameters α_1 and α_2 are weight coefficients which control how each distance, spatial d_{ij}

and temporal T_ij, influence the spatio-temporal distance. It should be pointed here that before using the equation (2.6), both values of d_{ij} and T_{ij} must be normalized by their maximum or minimum values.

We consider $[A_{start}, A_{end}]$ and $[B_{start}, B_{end}]$ the time windows of customer A and B respectively, with $A_{start} < B_{start}$. Temporal distance between the two time windows T_{ij} used in the equation above is defined and addresses three different scenarios:

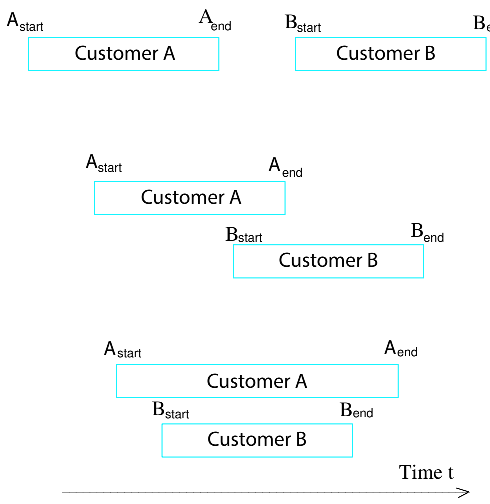

FIGURE 2.1: Time windows overlap scenarios

Three different situations can be considered according to the values of time windows. If $A_{end} < B_{start}$, there is therefore no overlap between the two time windows. If $A_{end} \geq B_{start}\ and\ A_{end} < B_{end}$, There is a partial overlap between the two time wiwdows. Finally, if $A_{start} \leq B_{start}\ and\ A_{end} \geq B_{end}$, then a total overlap occurs.

Based on the presented cases, when two costumers have overlapped time windows, they should be served in the same time. Then, the temporal distance between them is 0. Else, it can be determined as in Equation (2.8):

$$T_{ij} = \begin{cases} B_{start} - A_{end} & if\ A_{end} < B_{start} \\ 0 & if\ A_{end} \geq B_{start} \end{cases} \tag{2.8}$$

Description of our approach

Our methodology can be described on two steps. From one hand, the manner of clustering uses the K-medoid as a paradigm to tackle the pre-treatment process. On the other hand, the second step is devoted to select the adequate routes. This is the widespread ideas behind this approach. In the following, we provide more precise statements related to each step in more details:

Phase 1: It consists in identifying a set of clusters through a K-Medoid algorithm. The main idea of this iterative clustering algorithm is to divide the input data set into K distinct clusters $C_1, ..., C_K$.

Phase 2: It aims at finding a routing solution by solving each sub-problem related to each cluster. Then, we collect the solutions related to each sub-problem and we gather them to obtain a complete solution when the sub-solutions will be the routes of the final solution. For this purpose, we use three different meta-heuristics separately namely ALNS, VNS and GA in order to validate the results obtained in phase 1.

Clustering algorithms

K-Medoid algorithm

K-Medoid is an iterative clustering algorithm which aims to divide the data set into K pre-defined distinct non overlapping clusters as such manner that the group similarity between cluster center points and data set point will be maximized, and the similarity distance between groups will be minimized. In this work, we measure the cluster similarity by the presented spatiotemporal distance between cluster medoids and data-set point. The general concept of this clustering algorithm can be outlined in the following steps:

- ◇ Select K of the N input data points as the initial medoids.
- ◇ Associate each data point to the closest medoid x by computing the spatio-temporal distance.
- ◇ Define the y point coincidence.
- ◇ If swapping x and y minimizes the cost function, swap x and y.
- ◇ Repeat the three previous steps until there is no change in the assignments.

K-Means algorithm

The goal of this algorithm is to identify first, K pre-defined distinct non overlapping

set of clusters in order to maximize the group similarity measured by a specific similarity distance between cluster center points and data set point, and to minimize this cluster similarity distance between groups.
The basic idea of this iterative clustering algorithm can be summarized in the following steps:

- ◇ Choose the number of clusters K.
- ◇ Randomly generate K cluster centroïds (cluster centers).
- ◇ Compute the squared Euclidean distance between centroïds and data set point and assign each point to nearest cluster centers.
- ◇ Recompute the position of new centroïds by taking the average of the all data points that belong to each cluster.
- ◇ Repeat the two previous steps until some convergence criterion is met or the center points are not changed.

Routing algorithms

Initial solution
In this section, we used the greedy insertion heuristic introduced by (Solomon 1987) in order to generate the initial solution for the process of the routing metaheuristics. This method consists of finding the best location of a given node by testing the different possible configurations. more explicitly, the algorithm selects the best feasible insertion place in the current route for each non inserted node considering two factors: the increase in total cost of the current route after the insertion, and the delay of service start time of the client following the new inserted client. This process ends when all deleted nodes will be inserted. The detailed algorithm of the greedy insertion is shown bellow, while the detail of findClosest function used in this algorithm is given in the Appendix.

Algorithm 3 Greedy insertion operator

```
Inputs: solution $x$, Set $remainingNodes$
Outputs: solution $x$
Build a new route $cur$
Add $cur$ to $routes(x)$
while $remainingNodes \neq null$ do
  $nodeId = findClosest(remainingNodes)$
  if $nodeId \in remain$ then
    Remove $nodeId$ from $remain$
    Add $nodeId$ to $cur$
  else
    $cur = newroute$
    Add $cur$ to $routes(x)$
  end if
end while
return $x$
```

Variable Neighborhood Search

VNS, developed by Mladenović and Hansen, 1997, is a metaheuristic method which systematically changes the neighborhood in order to improve the incumbent solution and to avoid getting stuck in a local optimum within limited research area. The VNS algorithm consists in iteratively shaking a current solution in order to perform local search on it, and therefore deciding whether to reject or to accept it as a solution.

The process starts by constructing an initial solution x and defining a set of different neighborhoods $\{N_1, N_2, ..., N_k\}$. Then, we generate a random neighbor x' from the k^{th} neighborhood $N_k(x)$ of x, and we perform a local search in order to improve the solution to x'. The obtained solution will be accepted only if it improves the best found one x^b, and the neighborhood will be reset to N_1. Otherwise, we continue to search with the next neighborhood N_{k+1}.

These steps are summarized in the following algorithm:

Algorithm 4 Variable Neighborhood Search

```
Input: Initial solution $x$
Parameters: Shaking neighborhoods $\{N_1, N_2, ..., N_k\}$
$x^b = x$
**repeat**
  $k = 1$
  **repeat**
    Generate a random neighborhood $x'$ from the $k^{th}$ neighborhood $N_k(x)$ of $x$
    $x'' = localsearch(x')$
    **if** $f(x'') < f(x^b)$ **then**
      $x^b = x''$
      Continue to search in $N_1$
      $k = 1$
    **else**
      $k = k + 1$
      Move to a new neighborhood area
    **end if**
  **until** $k = k_{max}$
**until** Stop criteria is met
**return** $x^b$
```

It should be pointed out that the change of neighborhood is performed in a predefined order. In this section, four adjacent neighborhood structures were considered. The first operator N_1 is Two-opt. Its objective is to replace two arcs in a route by two others of the same route while reversing the route orientation. The second neighborhood structure N_2 is known as Relocate operator which changes the location of one node from one route to another. The third operator N_3 is referred to as Swap-move(k) and aims to the exchange a set of k-consecutive nodes in the original route by another without reversing the direction. The fourth operator N_4 is Exchange(k, k1) and consists on exchanging a value of k and k1 nodes between two distinct routes.

Genetic Algorithm

Genetic algorithm (GA) introduced by Prins, 2004, is a search method that adopt the natural evolution process, by operating on a set of initial solutions called "population of individuals". The GA generates solutions using techniques which are inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The GA outperforms all known metaheuristics that solves large-scale instances with high solution quality.

The process starts with a set of initial solutions called a Population of individuals. An individual is characterized by a set of parameters known as Genes. Genes are joined into a string to form a Chromosome (solution). For each individual, we calculate a fitness value which scores the ability of an individual to compete with other individuals. The probability that an individual will be selected for reproduction is based on its fitness score.

The next step, two pairs of individuals (parents) that have the highest fitness scores are selected in order to pass their genes to the next generation (offspring). To realize that two other steps are necessary. The first one is referred to as Crossover. It is a genetic operator used to exchange the genes of parents among themselves to generate new offspring. The second step called mutation occurs to maintain diversity within the population and prevent premature convergence.

This process will be repeated until the chance of producing off-springs which are significantly different from the previous generation is excessively low. The whole process is summarized in this algorithm:

Algorithm 5 Genetic Algorithm

```
Generate the initial population
Compute fitness
repeat
  Selection
  Crossover
  Mutation
  Compute fitness
until population has converged
```

Computational experiments

In this part, we summarize a few of the results obtained by evaluating different approaches conceived of to solve the vehicle routing problem with time windows.

The main purpose is to explore the effect of incorporating the K-Medoid with a spatio-temporal similarity distance in the clustering phase, on the objective value. From one hand, we attempt to solve the VRPTW problem by applying three different routing algorithms (i.e., Adaptive Large Neighborhood Search (ALNS), Variable Neighborhood Search (VNS) and Genetic Algorithm (GA)) separately in order to validate the results found in the first level, which leads to three different combinations. On the other hand, we compare our result to those achieved by using: the the K-Medoid with spatial distance, K-Means with spatial distance, and the K-Medoid with spatial distance.

Instance	ALNS	K-Means spatial	K-Means spatio-temporal	K-Medoid spatial	K-Medoid spatio-temporal
R101	1690.88	1684.87	1645.79	1657.07	1645.79
C101	843.12	828.94	828.94	828.94	828.94
RC101	1719.21	1713.22	1696.94	1708.90	1696.94
R201	1281.86	1274.93	1262.39	1274.93	1252.37
C201	632.99	613.64	598.15	603.41	591.56
RC201	1451.80	1442.99	1424.09	1431.66	1406.94
R121	4841.54	4825.10	4793.37	4803.18	4784.11
C121	2751.48	2743.49	2704.57	2719.52	2704.57
RC121	3622.97	3619.82	3602.80	3608.89	3602.80
R221	4502.62	4502.62	4483.16	4483.16	4483.16
C221	1942.06	1942.06	1931.44	1931.44	1931.44
RC221	3192.17	3163.35	3109.12	3130.10	3099.53
R141	10395.63	10395.63	10372.31	10381.74	10372.31
C141	7249.33	7194.13	7189.10	7190.64	7181.04
RC141	8608.29	8597.29	8572.36	8583.37	8572.36
R241	9270.33	9251.73	9216.86	9227.05	9210.15
C241	4195.50	4163.53	4132.61	4139.13	4123.07
RC241	6704.02	6704.02	6693.25	6698.54	6682.37
R161	21432.49	21432.49	21413.13	21432.49	21409.13
C161	14107.77	14095.64	14095.64	14095.64	14095.64
RC161	17042.94	17033.51	16997.31	17033.51	16997.37
R261	18261.11	18243.24	18205.85	18215.32	18205.85
C261	7860.43	7860.43	7809.52	7834.43	7783.03
RC261	13392.40	13360.07	13324.93	13341.83	13324.93
R181	36820.51	36805.15	36781.24	36792.12	36781.24
C181	25131.76	25106.51	25030.36	25093.27	25030.36
RC181	30529.18	30516.49	30481.37	30489.03	30464.65
R281	28231.09	28207.63	28162.14	28184.14	28137.27
C281	11749.77	11731.05	11703.95	11724.22	11684.43
RC281	21103.49	20981.14	20981.14	20981.14	20981.14
R1101	53489.88	53473.40	53451.17	53468.17	53439.07
C1101	42530.20	42521.27	42492.41	42508.20	42478.95
RC1101	45891.61	45876.32	45845.09	45863.54	45830.62
R2101	42261.73	42203.40	42182.57	42197.87	42182.57
C2101	16990.12	16919.68	16903.12	16912.02	16892.78
RC2101	30340.02	30297.92	30284.15	30290.23	30276.27

TABLE 2.3: Comparison of objective function between the different approaches using ALNS

Instance	VNS	K-Means spatial	K-Means spatio-temporal	K-Medoid spatial	K-Medoid spatio-temporal
R101	1690.88	1684.87	1657.07	1684.87	1657.07
C101	855.76	828.94	828.94	828.94	828.94
RC101	1758.10	1745.48	1619.21	1738.62	1713.22
R201	1281.86	1274.93	1262.39	1274.93	1262.39
C201	661.65	632.99	613.64	632.99	591.56
RC201	1503.13	1481.94	1424.09	1451.80	1417.57
R121	4862.21	4841.54	4793.37	4825.10	4784.11
C121	2770.35	2751.48	2719.52	2751.48	2719.52
RC121	3643.18	3634.37	3619.82	3622.97	3608.89
R221	4523.48	4502.62	4483.16	4502.62	4483.16
C221	1942.06	1942.06	1931.44	1931.44	1931.44
RC221	3192.17	3163.35	3109.12	3130.10	3102.06
R141	10451.07	10395.63	10381.74	10395.63	10372.31
C141	7275.09	7249.33	7194.13	7194.13	7152.02
RC141	8608.29	8597.29	8572.36	8583.37	8572.36
R241	9270.33	9251.73	9216.86	9227.05	9210.15
C241	4230.12	4195.50	4139.13	4163.53	4132.61
RC241	6742.31	6704.02	6682.37	6698.54	6682.37
R161	21503.69	21462.07	21432.49	21462.07	21432.49
C161	14165.36	14137.52	14107.77	14123.49	14095.64
RC161	17091.59	17042.94	17033.51	17042.94	17033.51
R261	18346.21	18261.11	18215.32	18243.24	18205.85
C261	8078.10	7913.28	7809.52	7860.93	7800.33
RC261	13471.79	13392.40	13324.93	13360.07	13324.93
R181	36988.61	36872.31	36805.15	36820.51	36785.07
C181	25131.76	25106.51	25030.36	25093.27	25030.36
RC181	30631.09	30529.18	30464.65	30516.49	30464.65
R281	28231.09	28207.63	28162.14	28184.14	28162.14
C281	11804.51	11762.80	11731.05	11749.77	11724.22
RC281	21180.64	21103.49	20981.14	21062.13	20981.14
R1101	53560.15	53512.26	53483.40	53489.88	53451.17
C1101	42530.20	42521.27	42478.95	42521.27	42478.95
RC1101	46021.16	45930.42	45876.32	45891.61	45845.09
R2101	42383.32	42261.73	42182.57	42203.40	42182.57
C2101	17131.45	16990.12	16900.80	16919.68	16900.80
RC2101	30399.74	30340.02	30284.15	30297.92	30276.27

TABLE 2.4: Comparison of objective function between the different approaches using VNS

Instance	GA	K-Means spatial	K-Means spatio-temporal	K-Medoid spatial	K-Medoid spatio-temporal
R101	1672.42	1672.42	1645.79	1652.28	1645.79
C101	837.16	828.94	828.94	828.94	828.94
RC101	1699.10	1699.10	1699.10	1697.34	1696.94
R201	1271.70	1271.70	1252.37	1260.28	1252.37
C201	618.88	618.88	597.04	608.34	591.56
RC201	1426.09	1406.94	1406.94	1406.94	1406.94
R121	4791.05	4799.18	4784.11	4793.19	4784.11
C121	2712.10	2735.70	2712.10	2715.93	2704.57
RC121	3619.82	3619.82	3602.80	3608.89	3602.80
R221	4523.70	4508.13	4489.37	4497.14	4483.16
C221	1961.19	1934.21	1931.44	1934.21	1931.44
RC221	3142.66	3121.67	3099.53	3106.09	3099.53
R141	10399.33	10395.63	10372.31	10381.74	10372.31
C141	7194.05	7189.38	7174.75	7179.24	7168.11
RC141	8599.17	8588.44	8577.12	8577.12	8572.36
R241	9270.33	9234.29	9216.86	9221.54	9210.15
C241	4163.53	4143.13	4123.30	4137.41	4116.05
RC241	6749.44	6709.71	6690.52	6697.78	6682.37
R161	21429.54	21423.85	21413.13	21423.85	21409.13
C161	14095.64	14095.64	14095.64	14095.64	14095.64
RC161	16994.24	16994.24	16986.26	16994.24	16982.86
R261	18250.38	18216.04	18212.04	18216.04	18205.85
C261	7833.32	7816.24	7799.62	7804.37	7783.03
RC261	13376.27	13350.16	13332.17	13338.63	13324.93
R181	36802.22	36779.14	36769.51	36772.79	36769.51
C181	25091.93	25072.34	25049.30	25061.74	25030.36
RC181	30516.73	30488.17	30470.61	30482.38	30464.65
R281	28151.64	28151.64	28136.15	28143.84	28132.80
C281	11692.73	11692.73	11683.04	11689.45	11679.63
RC281	21033.30	21012.36	21001.02	21012.36	20981.14
R1101	53461.04	53439.41	53420.13	53430.50	53412.11
C1101	42506.56	42491.84	42478.95	42478.95	42478.95
RC1101	45863.96	45863.96	45841.16	45849.46	45830.62
R2101	42213.75	42189.15	42182.57	42182.57	42182.57
C2101	16904.84	16904.84	16879.24	16886.12	16879.24
RC2101	30331.51	30315.95	30295.12	30304.17	30276.27

TABLE 2.5: Comparison of objective function between the different approaches using GA

It is clear from the tables (2.3, 2.4, and 2.5) above, that the approach which uses the k-Medoid clustering algorithm combined with the spatio-temporal similarity distance, considerably improves the quality of the solution, and leads to solutions that have lower route costs no matter what metaheuristics are used for routing. The power of this combination can be explained by the fact that K-Medoid is more robust to noise and outliers and it is more flexible to be used with any similarity measure in contrast of other partitioning techniques which are not sensitive to noisy data. Besides, the spatio-temporal measure seeks to explore the spatial similarity and temporal similarity between customers in terms of both the travel distance and time windows aspects. Thus, it assigns the customers which have a close spatial distance without missing opportunities to serve the customers which have a close time windows of service.

2.4 Thread Parallelism of Adaptive Large Neighborhood Search

In this section, we provide a new approach of parallel adaptive large neighborhood search algorithm designed to solve the vehicle routing problem with time windows. The main challenge is to obtain a reduced real-time solution without compromising the quality of the solution specially for large instances. Our main contribution introduces a procedure for the Thread parallelism of greedy insertion used in the initial block and also the destroy/repair operators involved in process of the ALNS metaheuristic. For this cheapest insertion Solomon's greedy heuristic, we refer the reader to the paper of (Bräysy and Gendreau, 2005b).

As an improvement of our approach, we use an iterative clustering algorithm as pre-processing step to perform initial data and to affect them to the worker thread. We use K-Means for the clustering algorithm as prototype which it is not restrictive method but of course, we can adopt other techniques such as K-Medoids or density based spatial clustering of applications with noise, see for e.g. Cömert et al., 2017. The computational results are performed such a way to find a compromise between the quality of the solution and the time of execution.

2.4.1 Motivation of our parallel ALNS

In this subsection, we will give some insights of the motivation of our parallel approach of the Adaptive Large Neighborhood Search ALNS proposed by Pisinger and Ropke, 2007 as an extension of the original work of Shaw, 1998 of the heuristic large neighborhood. The idea behind ALNS is to explore the solution space using many large neighbourhoods. At each iteration we choose which one to explore, based on a score that reflects its past performance. The neighbourhood to be chosen must have the higher score. This is possible by the application of destroy and repair functions. The idea is to define the approach in which the search is performed. More explicitly, the destroy procedure removes requests from the solution and the repair procedure reinsert them at more favored position while respecting the priority constraints.

The first challenge of such method is to optimize the primary time needed for the research and hence to explore a neighbourhood efficiently. In fact, the initialization block succeeded by the repair and destroy blocks of the sequential ALNS consumes most of execution time compared to other blocks because it is responsible for inserting all the nodes at the beginning of the process. In the contrast of the roulette wheel selection Pisinger and Ropke, 2007 which has a minimal execution time, since it does not contribute to the improvement of the current solution but it deals with the choice of the heuristic to apply. The destroy and repair operators are selected by a roulette wheel mechanism which controls the choice of the operators, with adaptive probabilities that depends on their past performance to decide which neighbourhoods to use. Every operator is associated with a score and a weight. Initially, all weights are set to one and all scores are set to zeros. Operators that have successfully found new improving solutions have a higher score and therefore a higher probability. Note that the destroy and repair neighbourhoods are selected independently, and hence two separate roulette wheel selections are performed. The scheme used for adjusting the weights and scores in the roulette wheel principle is described in the paper of Pisinger and Ropke, 2007.

Table 2.6 depicts our observation and confirms our assertion that the primary time of the initialization block consumes more time compared to the roulette, destroy or repair blocks as far as can be seen. This is the reason of our motivation to use the parallel algorithm in the initialization stage as described in more detail in the next section.

Instance size	Initialization block	Roulette block/iteration	Repair block/iteration	Destroy block/iteration
100	20	6	15	11
500	97	6	20	17
1000	152	8	48	40
2000	197	11	112	89
3000	316	19	184	136
4000	891	19	415	382
5000	1290	10	583	453
6000	1925	20	1090	994
7000	2658	13	1627	1240
8000	3546	17	1860	1410

TABLE 2.6: The execution time of the sequential ALNS blocks in ms

2.4.2 Description of our new approach

In this section, we start with a general exposition of our new parallel adaptive large neighborhood search (PALNS) algorithm used in the present work before to describe the detail of different components. The setup adopted here is slightly different from parallel variants of the large neighborhood search algorithm proposed for the Traveling Salesman with pickup and delivery and the capacitated VRP (Røpke, 2009), the Technician Routing and Scheduling Problem TRSP (Pillac et al., 2013) and the Periodic Location Routing Problem PLRP (Hemmelmayer, 2014)).

To understand the ideas behind our approach and the difference with previous works, we propose a simplified presentation with three levels developed in one or more algorithms illustrated in Figures: (Fig. 2.2) and (Fig. 2.3). For the sake of clarity, we focus mainly on the comparison of our contribution (Fig. 2.2) and others approaches (Fig. 2.3) at each stage with special emphasis on the additional used steps.

At the first level or the initialization phase in the previous works (Fig. 2.3), the initial solution is generated by assigning each customer a random combination. Instead, our method uses a parallel greedy insertion to produce an initial feasible solution (Fig. 2.2). As detailed in the next section, the principle of this greedy insertion is to select the best feasible insertion place in the current route for each non inserted node considering two factors: the increase in total cost of the current route after the insertion, and the delay of service start time of the client following the new inserted client.

The second level in the previous works (Fig. 2.3) is related to parallel destroy and repair operators assigned to a set of covering problems with routes taken from initial solution. Each thread worker improve solutions to the problem instance and write a single file for each route discovering during the run of the metaheuristic destroy and repair algorithms. Then the third level collect the set of routes of different local solutions obtained by each thread in attempt to combine it into new better temporary global solution and send it to be improved. At this stage, the solution will be accepted or rejected according to an acceptance criteria. A simulated annealing is the most used criterion as subprocess in the previous approach. The shared current and global best solutions are updated as necessary by repeating the process until a stop criterion is met.

We should point out here that the ALNS uses an adaptive layer with a set of removal and insertion heuristics and applies them by a selection mechanism preference that considers the statistics obtained during the search. On the other hand, the LNS heuristic doesn't use this scoring mechanism.

In contrast to the second and third levels of the previous works and instead of performing destroy and repair operators in each worker thread, we split the process in our approach, see Fig. 2.2, in four steps. The first step is devoted to removing a fixed number of nodes such that each thread contributes to rip up a part of the current initial solution obtained during the greedy insertion processing by using as we mentioned already three layers of destruction operators. After a complete run of this metaheuristic algorithm, we aggregate in the next processing step the formed destroy solution and send it back to the master processor. In the meanwhile, each thread worker reads different route file and solve set covering problem instances by repairing the solution based on the Solomon's insertion heuristic which we will explain later in more detail in the next section. In the last step, we combine routes from different solutions obtained by distinct threads into repaired solution. At the end, a new solution is accepted or rejected according to an acceptance criteria of Hill Climbing type. This process will be repeated until a stop criterion is met. We point out that Hill climbing is at the extreme of the spectrum of acceptance criteria, which only accepts solutions that improve on the current one. For a complete description of the different acceptance criteria tested within the ALNS framework including Hill Climbing type, we refer the reader to recent study of Santini, Ropke, and Hvattum, 2018 and references therein.

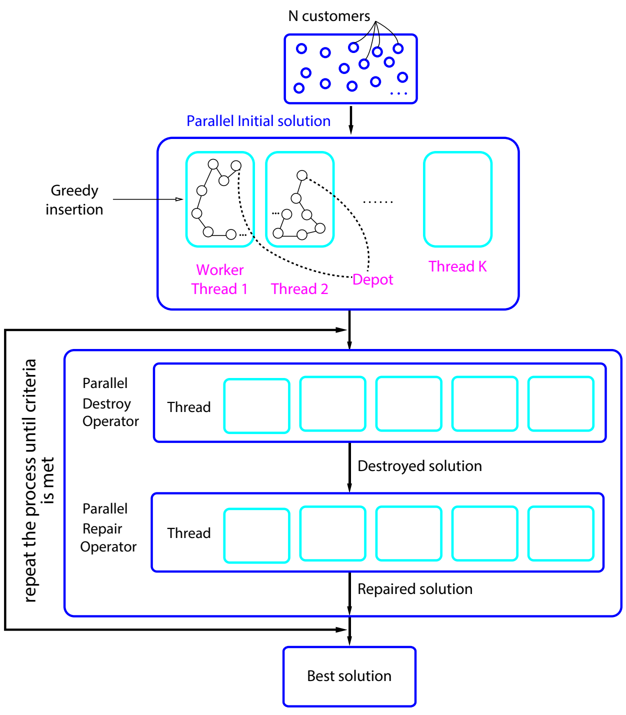

FIGURE 2.2: Our parallel Adaptive Large Neighborhood Search Algorithm

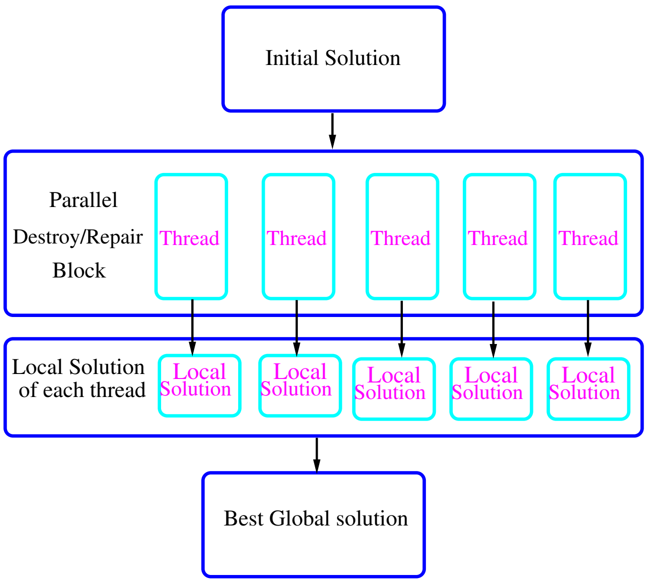

FIGURE 2.3: Previous works related to the parallel ALNS

The detail of our PALNS algorithm is the following:

Algorithm 6 Parallel adaptive large neighborhood search

```
1: Input: a parallel feasible solution $x$
2: Shared data: repair neighborhood $\rho^+$, destroy neighborhood $\rho^-$
3: Parallel initialization: $x^b = x$, $\rho^+ = 1$ and $\rho^- = 1$
4: repeat
5:     Choose a destroy neighborhood $\rho^-$ using roulette wheel selection based on previously obtained scores $\pi_j$
6:     Apply the parallel destroy heuristic corresponding to the destroy neighborhood $d(x)$
7:     Choose a repair neighborhood $\rho^+$ using roulette wheel selection based on previously obtained scores $\pi_j$
8:     Apply the parallel repair heuristic corresponding to the repair neighborhood $r(d(x))$
9:     if $r(d(x))$ can be accepted then
10:        $x = r(d(x))$
11:    end if
12:    if $c(r(d(x))) < c(x)$ then
13:        $x^b = r(d(x))$
14:    end if
15:    Update scores $\pi_j$ of $d$ and $r$
16: until Stop criteria is met
17: return $x^b$
```

Initial solution

The ALNS method requires an initial solution for its process. Commonly, we use a greedy approach to initialise and to improve the solutions by applying an algorithm with large neighbourhood exchanges. Inspired by the sequential greedy algorithm presented by Shaw, 1997, we develop a parallel greedy insertion heuristic where the pseudo code is shown in following algorithm:

Algorithm 7 Parallel greedy insertion operator

```
Inputs: solution $x$, Set $remainingNodes$, Set $remain1$, $remain2$, $remain3$, $remain4$
Outputs: solution $x$
Divide the nodes of $remainingNodes$ on $remain1$, $remain2$, $remain3$ and $remain4$
for $remain \in \{remain1, remain2, remain3, remain4\}$ in parallel do
  Build a new route $cur$
  Add $cur$ to $routes(x)$
  while $remain \neq null$ do
    $nodeId = findClosest(remain)$
    if $nodeId \in remain$ then
      Remove $nodeId$ from $remain$
      Add $nodeId$ to $cur$
    else
      $cur = newroute$
      Add $cur$ to $routes(x)$
    end if
  end while
end for
return $x$
```

The input data is the set of nodes of the studied problem which we divided in four sub-problems. The nodes of sub-problem are assigned randomly to a worker thread as it is depicted in line 3. We construct a new route for each thread added to the list of the current solution routes in order to find the best closed location of a given node by testing different possible configurations in line 5-17. The rule of this selection depends mainly on the total cost of the current route after the insertion and the start service time of the client following the new inserted client. It is not necessary that the quality of the solution to be high because this solution will subsequently be improved using the ALNS method.

The algorithm performs parallel calculations that focus on minimizing insertion costs after each insertion. More precisely, it is a minimization of the difference in the value of the objective function before and after insertion, and then inserting the node in the least expensive position. First, we affect the non-inserted nodes to four threads in order to calculate the lowest insertion cost for their assigned nodes c_i, and afterward insert the one with the minimum cost. This process ends when all deleted nodes will be inserted. The algorithm of findClosest function used by the greedy insertion operator can be found in Annexe.

Solution destruction

In the destruction phase, three different parallel procedures is used to ensure the diversity of the searching process and to define the neighborhood to explore at each iteration. Each destroy method described below uses a defined number of threads to remove in parallel way a predefined number of nodes.

Parallel proximity operator

The objective of this operator is to select close clients i, j both geographically and temporally taking into account time windows constraints $a_l, b_l, \quad l = i, j$ and the non negative distance associated or travel time t_{ij}. Two close customers are compatible if and only if

$$a_i + t_{ij} \leq b_j \tag{2.9}$$

The proximity of time windows T_{ij} of two neighbor customers is defined as the width of the interval of the feasible visiting time at customer j when customer i is visited immediately before j

$$T_{ij} = min\{b_j, b_i + t_{ij}\} - max\{a_j, a_i + t_{ij}\} \tag{2.10}$$

First, we arbitrarily choose a client i from the list of customers that were removed, then all other clients j are assigned in decreasing order of a spatio-temporal proximity measure, as proposed by Shaw, 1998 and Shaw, 1997, to customer i. This measure is defined by

$$R(i,j) = \frac{1}{\frac{min\{c_{ij}, c_{ji}\}}{c_i^{max}} + \frac{1}{T_{ij}+T_{ji}}} \tag{2.11}$$

where c_i^{max} is the greatest cost for each customer i not pulled back. Clients to be removed are then selected, choosing those with a higher spatial-temporal measure. The detail of our algorithm called parallel proximity operator is given below:

Algorithm 8 Parallel proximity operator

```
Select randomly a node $firstToRemove$ from the list of clients that have been re-
moved, and add it to the list $removedNodes$
Add the other nodes to $remainingNodes$
**for** $i = 1$ $to$ $numToRelax$ in **parallel do**
    Choose randomly a node $removedNodeId$ from $removedNodes$
    Choose by Rank and relatedness a node $nodeId$
    Add $nodeId$ to $removedNodes$
    Remove $nodeId$ from $remainingNodes$
**end for**
**return** $removedNodes$
```

Parallel route portion operator

The parallel proximity operator presented above could remove only one client by road, giving no flexibility of change on these roads, as a substitute procedure, the route portion operator choose a pivot client owned to a road and remove it as well as its adjacent clients.

Every one of the other nodes will be assigned to a thread in order to calculate the spatio-temporal measure for each one, see for instance Shaw, 1998, with the objective to select a second client belonging to another route but close to the initial pivot. This second pivot is removed from the solution as well as its adjacent clients and so on until all clients will be removed. Since there can only be one pivot per road, the adjacent customers of the pivot which will be removed on each side are chosen by a ratio at a maximum distance d:

$$d = f_{rp} \; max\{c_{ij}, c_{jk}\} \tag{2.12}$$

by taking j as the pivot, and i and k are their immediate adjacent customers in the route and the multiplier f_{rp} changes during the iterations, its initial value is 1. This multiplier guarantees that we choose the ideal number of customers to remove.

The description of the algorithm is given below. For simplicity purposes, the detail of the first and second lines is omitted but from the definition of the maximum distance d given in the equation (2.12), the procedure is clear.

Algorithm 9 Parallel route portion operator

```
Choose the first node to delete and its adjacents and add them to $adjacents$ list
Add the nodes of $adjacents$ to $removedNodes$ list and remove them from $remainingNodes$
Add the other nodes to $remainingNodes$
**for** $i = 1$ $to$ $numToRelax$ in **parallel do**
   Choose randomly a node $removedNodeId$ from $removedNodes$
   Choose by Rank and relatedness a node $nodeId$
   Add $nodeId$ to $removedNodes$
   Remove $nodeId$ from $remainingNodes$
**end for**
**return** $removedNodes$
```

Parallel longest detour operator

This operator deletes the customers that cause the largest increase in the cost of the current solution. We start by affecting all clients to the threads to sort them out in descending order by the detour they generate in the current solution. Each thread calculate for each client j served between i and k (customers or depot), the related detour called m in our algorithm below at line 4 defined as $c_{ij} + c_{jk} - c_{ik}$. We removed then the node which generate a maximum detour called z, line 7. For more details, we refer the reader to (PrescottGagnon, Desaulniers, and Rousseau, 2009) and references therein.

Algorithm 10 Parallel longest detour operator

```
**Inputs:** feasible solution $x$, Array $maxTab$
**Outputs:** $z$ max detour
**for** $route \in routes(x)$ in **parallel do**
  $m = MaxDetour$
  Add $m$ to $maxTab$
**end for**
$z = max(maxTab)$
**return** $z$
```

Solution reconstruction

Solomon's insertion heuristic (1987) provided a method of selecting the new customer to be inserted into a route using two criteria $c_1(i, u, j)$ and $c_2(i, u, j)$ to select customer u for insertion between adjacent clients i and j in the current partial route. The first criterion c_1 is applied to calculate the best feasible insertion place in the current route for each unrouted client u as

$$c_1(i(u), u, j(u)) = \min_{\rho=1,\ldots,m} c_1(i_{\rho-1}, u, i_\rho) \tag{2.13}$$

The second criterion c_2 is then applied to select the new inserted customer:

$$c_2(i(u^\star), u^\star, j(u^\star)) = \max_u \{c_2(i(u), u, j(u)) | u \textit{ is unrouted and route is feasible}\} \tag{2.14}$$

Customer $u^\star$ is then inserted into the route between $i(u^\star)$ and $j(u^\star)$. The measurement of an insertion place c_1 is based on two factors: the increase in total distance of the current route after the insertion (c_{11}), and the delay of service start time of the customer following the new inserted customer (c_{12}). To be more precise, $c_1(i, u, j)$ is calculated as:

$$c_1(i, u, j) = \alpha_1\, c_{11}(i, u, j) + \alpha_2\, c_{12}(i, u, j),$$

$$\alpha_1 + \alpha_2 = 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0,$$

$$c_{11}(i, u, j) = d_{iu} + d_{uj} - \mu\, d_{ij}, \mu \geq 0,$$

$$c_{12}(i,u,j) = b_{ju} - b_j.$$

where $d_{iu} + d_{uj}$ is the new distance between two nodes i and j after the insertion, b_j is the previous service start time, d_{ij} is the old distance between i and j and b_{ju} is the new service start time of customer j after the insertion of customer u.
The criterion $c_2(i,u,j)$ is calculated as following

$$c_2(i,u,j) = \lambda \, d_{0u} - c_1(i,u,j), \quad \lambda \geq 0, \tag{2.15}$$

where the parameter λ is used to define how much the best insertion place for an unrouted customer depends on its distance from the depot and extra time required to visit the customer by the current vehicle.

2.4.3 $K-$means

In order to improve the quality of the parallel solution, a preprocessing clustering analysis of $K-$means is introduced . The goal at this stage is to identify first, K pre-defined distinct non overlapping set of clusters and to assign them to K worker threads in order to maximize the group similarity measured by the average squared Euclidian distance between cluster center points and data set point, and to minimize this cluster similarity distance between groups.
The basic idea of this iterative clustering algorithm can be summarized in the following steps:

- ◇ Choose the number of clusters K.
- ◇ Randomly generate K cluster centroïds (cluster centers).
- ◇ Compute the squared Euclidian distance between centroïds and data set point and assign each point to nearest cluster centers.
- ◇ Recompute the position of new centroïds by taking the average of the all data points that belong to each cluster.
- ◇ Repeat the two previous steps until some convergence criterion is met or the center points are not changed.

2.4.4 Computational results

In order to examine the performance of the suggested version of parallel ALNS, we accomplished various computational tests. It is important to mention that the notion of thread parallelism used in our context can be defined as the capability of a processing unit to execute multiple processes contemporaneously or with time slicing. By means of thread, the smallest unit of processing that can be performed in an operating systems in order to accelerate the execution time and manage the code over time.

The Thread parallelism approach was tested on a set of small instances based on the reference of benchmark, and its extension the instances of Gehring and Homberger's benchmark. Notwithstanding, instances larger than 1000 are generated randomly so that the euclidian distance between each pairs of nodes goes from 0 to 100, the same thing for service time. The time interval is of capacity 200 between the start and the end of the operations at each node. The solutions are obtained over the average of 10 trials. The algorithm was implemented in Java 7, compiled with Intel compiler Celeron 1.80 GHz core i5 with 8GB RAM. The parallel approach was run for 15600 iterations and for stopping condition of 20 minutes as limit time. The set-covering model was running 10 times for each instance. The datasets used for the present analysis of the VRPTW were generated randomly. The number of expected customers was chosen to be 100 as in Solomon's R101 dataset. The geographical locations of the customers were uniformly distributed in the square $[0, 100] \times [0, 100]$, with the depot located at the median, i.e. at $(50, 50)$. The vehicle speed was set such that 1 geographical unit could be driven in 1 minute. We used a constant service time of 10 minutes per customer.

The description of the impact of our approach is given following different parameters: execution time and objective function. The discussion about the best good solution of our VRPTW problem in reduced running time is the object of the last part of this section. In fact, even if the quality of the solution in this new approach can be improved, the introduction of K-means in the pre-processing step ameliorates considerably the quality of the solution. It should be pointed here, for the best of our knowledge, that all the numerical simulations of the previous works mentioned in this section are applied to different problems such as Capacitated VRP, Periodic Vehicle Routing Problem and TSP and not to the VRPTW subject of the this dissertation. We adopted only here the spirit of the previous works to the studied problem to perform new results.

Execution time

Table 2.7 presents the execution time of our new approach for different instance size depending on the number of threads in use. The choice of four threads is not restrictive and we can use threads as much as possible. The conclusion from this table is clear: more threads lead to the improvement of the average execution time.
Table 2.8 presents a comparison of the running time for each instance group of sequential ALNS, previous works and our approach with maximal number of iterations of 15600. As expected, the results show that the execution time is decreasing as the degree of parallelization is growing. The thread parallelism approach improves the average running time compared to the previous works and the difference becomes more significant where more threads are employed.
Table 2.9 reports for the group of instances 3000-8000 the comparison of different approaches the number of attainted iterations when the forcing stop condition is about 20 minutes. Our approach compared to the method used for the previous works reached more iterations for the same time interval. Thus may ameliorates the quality of our solution because the solution have more chance to escape from a local minimum. As far as we are aware, the maximal number of iterations for large instances is failing in the literature.

Instance size	1 thread	2 threads	3 threads	4 threads
10	24	23	19	5
50	26	24	21	8
100	28	25	23	15
200	34	29	24	18
400	45	40	30	24
600	48	45	34	28
800	54	51	48	37
1000	84	67	62	50
1500	144	104	76	70
2000	219	137	120	117
2500	315	249	195	142
3000	393	271	234	206
3500	589	302	276	251
4000	819	580	350	305
4500	936	637	416	380
5000	1049	667	471	421
5500	1349	756	529	446
6000	1954	989	667	570
6500	2601	1100	797	643
7000	2710	1210	874	702
7500	2992	1412	924	781
8000	3314	1547	985	816

TABLE 2.7: The average of execution time of an ALNS iteration in (ms)

Instance size	Seq	Previous works			Our approach		
		2 thread	3 thread	4 thread	2 thread	3 thread	4 thread
100	423.24	397.21	376.08	308.29	390.14	358.34	234.91
200	516.38	489.11	414.64	337.27	452.23	374.22	280.85
400	697.12	663.89	503.61	429.15	624.97	468.92	374.07
600	746.64	729.97	617.60	510.79	702.57	530.10	436.90
800	837.06	802.16	786.28	663.60	785.41	745.36	577.68
1000	1347.51	1113.37	1058.51	891.12	1045.45	967.06	780.49
2000	3229.73	2610.02	2149.50	1915.72	2137.67	1872.86	1725.42

TABLE 2.8: The execution time of parallel ALNS in seconds for 15600 iterations

Instance size	Seq	Previous works			Our approach		
		2 thread	3 thread	4 thread	2 thread	3 thread	4 thread
3000	4058	4996	5886	7042	5984	6813	7860
4000	3471	4317	5123	6608	5201	6122	7335
5000	2147	3142	4069	5521	4375	5136	6348
6000	1714	2570	3352	4610	3385	4020	5175
7000	1043	2181	2873	3481	2835	3233	4025
8000	762	1743	2112	2669	2032	2563	2994

TABLE 2.9: Number of iterations reached in 20 minutes

Objective function

The table 2.10 presents the objective function of thread parallelism approach compared to sequential ALNS. We computed then the mean absolute deviation (MAD) which is the absolute difference between cost function of the thread parallelism and the sequential ALNS divided by the magnitude of the objective function in sequential ALNS. In fact, when we divide the clients on the threads in the insertion phase, we reduce the chances of falling on a good neighbor and we increase the objective function of our parallel approach. This assertion is confirmed from detailed results in the table 2.10. The solution quality of the sequential ALNS at the average absolute deviation, for the instances $100 - 1000$, is between 13 and 28 percent better than our approach. For large instances, the cost function of multithreading parallel and sequential ALNS solution converges. We can conclude that our multithreading parallel approach is efficient for large instances.

Instance size	Sequential ALNS	Parallel ALNS	Mean Absolute Deviation
100	2252	2883	0.28
500	9418	11208	0.19
1000	17502	19777	0.13
2000	34420	36829	0.07
3000	50817	53866	0.06
4000	67675	71728	0.06
5000	82937	86249	0.04
6000	99072	102040	0.03
7000	116911	119241	0.02
8000	132631	133950	0.01

TABLE 2.10: Sequential and multithreading parallel ALNS comparison according to the objective function

Comparison according to Solomon's Benchmark

This section reports computational results showing the efficiency of our main contribution tested on a classical set of benchmark problems Solomon, 1987 and Gehring and Homberger, 1999 with a short planning horizon:

- Set R include problems with randomized customer locations.
- Set C include problems with customers grouped in clusters.
- Set RC include problems with both clustered and randomized customer locations

Table 2.11 summarizes the results for each instance group. The first column defines the instance group, the second column contains the results of the sequential ALNS and the third and fourth columns our results respectively by using the techniques in the previous works and our main contribution for the thread parallelism approach.

For instance group of size between 100 and 400, the quality of the solution obtained by sequential ALNS and our results by techniques of the previous parallel variant of ALNS are respectively much better than the thread parallelism approach with a percentage gap of an average respectively between $3.7\% - 17\%$ and $0.7\% - 8.7\%$ but with longer runtime.

When we increase the instance size, the quality of the solution of our approach becomes more interesting. As matter of the fact, it is clear from the table 2.11 that for instance size in the interval range $600 - 1000$, our algorithm yields better results than the parallel methods of the previous works with a profit between 0.07% till 2% in terms of solution quality and the objective function of our thread parallelism approach converges to the sequential one in shorter run-time as mentioned in the next table. Nevertheless, we suggested to use the K-means as clustering to improve our objective function as explained in the next section.

Table 2.12 depicts our improvement results in execution time for all instance groups compared to the sequential ALNS and previous works. The running is much faster using our thread parallelism.

Instance	Seqential ALNS	Previous works	Thread parallelism
R101	1696	1813	1970
C101	847	934	990
RC101	1859	2000	2128
R201	1295	1387	1509
C201	608	672	713
RC201	1531	1653	1751
R121	4928	5070	5489
C121	2898	3074	3300
RC121	3636	3810	4079
R221	5842	6017	6490
C221	1958	2068	2232
RC221	3142	3290	3532
R141	10955	11239	11543
C141	7726	7965	8017
RC141	8916	9147	9262
R241	9689	9951	10125
C241	4458	4601	4636
RC241	7009	7212	7275
R161	21956	22351	22355
C161	14839	15269	15264
RC161	18645	18961	18943
R261	18806	19144	19129
C261	8139	8351	8342
RC261	14536	14827	14798
R181	37911	38555	38388
C181	26561	27251	27169
RC181	31047	31823	31522
R281	29124	29608	29357
C281	12303	12698	12537
RC281	21505	22135	21679
R1101	53941	56422	56011
C1101	43092	44772	44378
RC1101	46479	48617	48131
R2101	42983	44570	44187
C2101	17186	17908	17702
RC2101	30851	32085	31622

TABLE 2.11: Solomon's instance: Comparison of objective functions of different parallel algorithms

Instance	Seq ALNS	Previous works	our Parallel
R101	368	252	197
C101	325	205	159
RC101	351	222	169
R201	422	289	226
C201	365	231	179
RC201	415	263	200
R121	490	356	279
C121	435	273	232
RC121	454	314	255
R221	588	428	335
C221	505	317	270
RC221	530	367	298
R141	581	395	344
C141	560	363	317
RC141	501	310	256
R241	736	501	436
C241	703	435	359
RC241	718	466	407
R161	556	403	356
C161	680	554	455
RC161	619	462	401
R261	749	543	480
C261	817	666	547
RC261	788	588	511
R181	692	497	467
C181	708	583	498
RC181	697	514	474
R281	852	612	576
C281	937	732	659
RC281	917	676	619
R1101	884	522	469
C1101	910	599	514
RC1101	903	563	499
R2101	1318	779	701
C2101	1361	896	770
RC2101	1331	830	737

TABLE 2.12: Solomon's instance: Comparison of runtime in seconds of different parallel algorithms

Numerical Solution using K-means clustering

We introduce K-means as a preliminary process to perform our new approach for instances starting from 100 till 8000. The table 2.13 illustrates the execution times of K-means clustering versus instance size. We can observe clearly that this processing step doesn't consume a lot of time since at the average the run-time is about 1.91490 seconds with a maximum for large instances of 3.762 seconds. On the counterpart the quality of the solution may be improved considerably.
Table 2.14 depicts different approaches for each instance size between 100 and 8000. It should be noted here that the K-means was applied only on our thread parallelism approach and not on the techniques used for previous works. We conclude from the table that for instance above 1000, our new approach assembles a better solution by using K-means clustering as pre-processing with four threads compared to the previous works and the gap with sequential ALNS is about 1-2 percent. As the instances become larger, this gap decreases. For instances $100 - 500$, the gap is significant and starts from 12% and reaches 18% for instance 100. We recover this gap starting from instance 1000 up to 2.4%. We suggested then this pre-processing clustering as a good compromise between the quality of the solution and the execution time for large instance. For detailed study on the all possible pre-processing, it will be studied elsewhere.

Instance size	K-means execution time
100	147
500	448
1000	701
2000	1200
3000	1731
4000	2271
5000	2585
6000	2963
7000	3341
8000	3762

TABLE 2.13: The execution time of the K-means in (ms)

Instance	Sequential	Thread parallelism	Previous works	Average: MAP
100	2252	2657	2597	0.18
500	9418	10548	10461	0.12
1000	17502	18553	18723	0.06
2000	34420	35797	36451	0.04
3000	50817	52850	53549	0.03
4000	67675	69705	70737	0.02
5000	82937	84596	85476	0.02
6000	99072	101053	101998	0.02
7000	116911	118080	119101	0.01
8000	132631	133826	133941	0.01

TABLE 2.14: The objective function of ALNS parallel (previous and new approach) and sequential ALNS with k-means versus sequential ALNS

2.5 Conclusion

In this chapter, we address three different approaches to enhance the process of the Adaptive Large Neighborhood Search presented by Pisinger and Ropke, 2007 in order to solve the Vehicle Routing Problem with Time Windows.

In the first strategy, we presented a novel heuristic inspired by the Adaptive Large Neighborhood Search (ALNS) previously suggested by Røpke and Pisinger, 2006. The proposed heuristic uses the Modified Choice function (MCF) of Drake, Özcan, and Burke, 2012 as an elegant selection mechanism to favor the most successful operators instead of the roulette wheel selection. This general method is denoted Modified Adaptive Large Neighborhood Search (MALNS). The competiveness of our proposed method is demonstrated on the Solomon's benchmark and its extension the instances of Gehring and Homberger's benchmark. We conclude that our MALNS algorithm outperforms the classical ALNS in terms of solution quality.

In the second strategy, we presented A hierarchical approach consisting of two stages as "Cluster first - Route second" is proposed. In the first stage, customers are assigned to vehicles using three different clustering algorithms (i.e., K-means, K-medoids). In the second stage, we solve the VRPTW using three distinct routing algorithms (i.e., ALNS, GA and VNS). The experimental results show that the proposed hierarchical approach enables us to deal efficiently with a large size real problem and to solve it in a short time using heuristic methods.

Finally, we introduced a new multithreading parallel adaptive large neighborhood search to solve the well known optimization VRPTW problem. Our algorithm is shown to be able to obtain a good solution in a reasonable time without too much compromise on the quality of the solution. To ensure the quality commitment of the solution, K-means clustering algorithm is proposed. Other techniques such as K-medoïds or density based spatial clustering of applications with noise can be also used as pre-processing step. Moreover, the implementation of thread parallelism of the operators of our approach can easily be applied to other variants of the VRP problem. Compared to the best performing algorithm from the previous solution methods on parallel ALNS for large instances, our algorithm can outperforms in terms of solution quality in shorter time. We can also improved the quality by the introduction of pre-processing clustering algorithm. The computational experiments are both tested on the Solomon's benchmark instance of VRPTW. We conclude that our main algorithm produced a better results in shorter runtime in terms of solution quality for the instance size above 400.

The perspective of this work, is to study the influence of the preprocessing step by comparing different cluster algorithms following similarity pair of clusters by introducing the linkage metrics (Minkowski, Mahanalobis, Pearson correlation, ...) for measuring proximity between clusters. Our guess is that this study will lead to improve our thread parallelism approach. Also, we aim to apply the multithreading parallelization technique to the MALNS algorithm, in order to develop fast optimization procedure able of reacting to changes in problem information in real time. We believe that this study will lead to improve the execution time of the MALNS method.

Chapter 3

Robust Vehicle Routing Problem with Time Windows

3.1 State of the Art

Over the past few decades, the Vehicle Routing Problem (VRP) and its variants have been the subject of massive investigations in operations research. This fact is due to the importance of its applications in different domains, such as logistics, supply chain management, scheduling, inventory, finance, etc. The main purpose of the vehicle routing problem is to find a set of least cost routes, beginning and ending at a depot, that together cover a set of customers (see, e.g., Eksioglu, Vural, and Reisman, 2009; Braekers, Ramaekers, and Nieuwenhuyse, 2015; Vidal, Laporte, and Matl, 2019). In real-world applications, several operational constraints must be taken into account, as for example considering the travel and service times with time-window limitations Bräysy and Gendreau, 2005a. Then, the considered problem becomes the Vehicle Routing Problem with Time Windows (VRPTW) Kallehauge et al., 2005.

A challenging topic in solving the VRPTW problem consists of considering uncertain parameters. Different approaches have been proposed in order to handle uncertain events in a VRPTW, in demand, displacement time, and service time. From the literature, we distinguish between stochastic and robust approaches. The stochastic variant can be regarded as a methodology that aims at finding a near-best solution for the objective function responding to all uncertain events that are characterized by their probability distributions Dror and Trudeau, 1986; Dror, Laporte, and Louveaux, 1993; Gendreau, Laporte, and Séguin, 1996. On the other hand, the purpose of the robust approach is to find a solution that protects against the impact of data uncertainties, taking into consideration several technical criteria challenges such as the worst case, best case, min-max deviation, etc. The choice of a mathematical model of uncertain data is a crucial step to provide robust solutions. This kind of approach was the subject of a series of papers (see, e.g., Nasri et al., 2020a). In this context, Rouky et al., 2018 introduced the uncertainties to the travel times of locomotives and the transfer times of shuttles as a model of the Rail Shuttle Routing Problem (RSRP) at Le Havre port. The authors proposed the Robust Ant Colony Optimization (RACO) as an efficient technique to deal with the problem. In the same spirit, Wu, Hifi, and Bederina, 2017 proposed a robust model tested on a set of random instances for the vehicle routing problem with uncertain travel time to improve the robustness of the solution, which enhances its quality compared with the worst case in a majority of scenarios. For instance, we split the resolution methods of this last approach into two major categories: exact and heuristic methods.

Due to the NP-hard nature of such problems, we cannot expect to use exact methods for the resolution (see, e.g., Gutin and Punnen, 2002; Toth and Vigo, 2002). Indeed, the heuristics are solution methods that yield very good solutions in a limited time at the expense of ensuring the optimal solution. A generic class called the metaheuristic is used to exploit the best capabilities to achieve better solutions to solve a wide range of problems, since the mechanism to avoid getting trapped in local minima is present. In this regard, the literature covers a considerable number of metaheuristics conceived to solve the VRPTW such as Simulated Annealing (SA) (Chiang and Russell, 1996; Afifi, Dang, and Moukrim, 2013), Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997; Dhahri et al., 2016), Ant Colony Optimization Algorithm (ACO) (Gambardella, Taillard, and Agazzi, 1999; Tan, Zhuo, and Zhang, 2006), Genetic Algorithm (GA) Thangiah, 1995, and Tabu Search (TS) (Potvin et al., 1996; Taillard et al., 1997; Bräysy and Gendreau, 2002).

The need for parallel computing becomes inevitable despite the good results obtained with metaheuristics, due to the huge scale of the input data and the unexpected way of its change, which makes the objective function time-consuming. However, it is important to notice that the quality of the solution can be influenced. For instance, the major challenge is to find a parallelization strategy that solves larger problem instances in reasonable computing times and offers a consistently high level of performance over a wide variety of problem characteristics. In this context, several authors have proposed parallel techniques to tackle the combinatorial problems. Following these ideas, Bouthillier and Crainic, 2005 proposed a multi-thread parallel cooperative multi-search method founded on a solution warehouse strategy to deal with the deterministic VRPTW. Their method is based mainly on two cooperating classes of heuristics, namely tabu search and the evolutionary algorithms. In this regard, Røpke, 2009 applied a parallel ALNS to the traveling salesman problem with pickup and delivery and the capacitated vehicle routing problem such that each worker thread obtains a copy of the current solution and performs destroy and repair operations on its local copy in order to produce the best global solution. In the same spirit, Hemmelmayer, 2014) proposed a parallel variant of the Large Neighborhood Search (LNS) to solve the periodic location routing problem. On the same topic, Pillac et al., 2013 presented a parallel version of the ALNS by adding a set covering post-optimization model that combines the tours generated throughout the search to assemble a better solution. It is worth mentioning here that the longer a heuristic is run, the better the quality of the solution is. Our contribution in this work is to study the balance between the quality of the solution and the corresponding execution time. Therefore, we suggest through our investigation a compromise between the required running time and the objective function. This can be viewed as a multi-criteria optimization problem.

3.2 Problem statement

The mathematical formulation of the problem can be represented on a graph $G = (N, A)$, where $N = \{o, 1, ..., n\}$ is the set of nodes and $A = \{(i, j) : i, j \in A, i \neq j\}$ is the set of arcs. The node o represents the depot and each other node is affected to a customer i. Each arc (i, j) is assigned to the travel cost c_{ij} which, in general, is proportional to the travel time t_{ij} or the distance d_{ij} between i and j. For the rest of this chapter, we consider only the travel time cost t_{ij}. It is worth mentioning that this travel time t_{ij} is subject to uncertainty Δ_{ij}. The nominal service time is denoted by P_i^k for each vehicle k and node i within the time window $[a_i, b_i]$ and depends on uncertainty δ_i. According to the work of Nasri et al., 2020a, we identify the uncertainty sets related to these times by:

$$U_t = \{\widetilde{t} \in \mathbb{R}^{|A|} \ / \ \widetilde{t_{ij}} = t_{ij} + \Delta_{ij}\epsilon_{ij}, \sum_{(i,j)\in A} \epsilon_{ij} \leq \Gamma, 0 \leq \epsilon_{ij} \leq 1, \ \forall(i,j) \in A\}$$

and

$$U_P = \{\widetilde{P} \in \mathbb{R}^{|N|} \ / \ \widetilde{P_i} = P_i + \delta_i\omega_i, \sum_{i\in N}\omega_i \leq \Lambda, 0 \leq \omega_i \leq 1, \ \forall i \in N\}$$

We denote the subset of arcs which are dependent on uncertainty by Ψ with a cardinal Γ and the subset of nodes depending on uncertainty by θ with a cardinal Λ. The binary decision variables x_{ij}^k take the value 1 if vehicle k travels between the pairs of nodes (i, j) and 0 otherwise.

We introduce the model of our problem which tries to find a solution optimizing the total travel time taking into account the minimization of the worst evaluation over all scenarios:

$$\textbf{Min} \ (\sum_{k\in V}\sum_{(i,j)\in A} x_{ij}^k t_{ij} + \max_{\{\Psi/\Psi\subset A, |\Psi|=\Gamma\}} \sum_{k\in V}\sum_{(i,j)\in\Psi} x_{ij}^k \Delta_{ij})$$

subject to

$$\sum_{k\in V}\sum_{j\in N} x_{ij}^k = 1 \qquad \forall(i \in N) \tag{3.1}$$

$$\sum_{j\in N} x_{0j}^k = 1 \qquad \forall(k \in V) \tag{3.2}$$

$$\sum_{i\in N} x_{ih}^k = \sum_{j\in N} x_{hj}^k \qquad \forall(h \in N)\ \forall(k \in V) \tag{3.3}$$

$$\sum_{i\in N} x_{i0}^k = 1 \qquad \forall(k \in V) \tag{3.4}$$

$$a_i \leq P_i^k \leq b_i \quad \forall(i \in N) \qquad\qquad \forall(k \in V) \tag{3.5}$$

$$P_i^k + t_{ij} + \delta_i \nu_i^\theta + \Delta_{ij}\mu_{ij}^\Psi - P_j^k \leq (1 - x_{ij}^k)M, \tag{3.6}$$

$$\forall(i \in N),\ \forall(j \in N \setminus \{0\}),\ \forall(k \in V)\ \forall(\theta \subset N) \mid \theta \mid= \Lambda,\ \forall(\Psi \subset A) \mid \Psi \mid= \Gamma$$

where M is a great value, ν_i^θ and μ_{ij}^Ψ are two indicator functions. When $i \in \theta$, ν_i^θ takes the value 1. And when $(i, j) \in \Psi$, μ_{ij}^Ψ takes 1.

The constraint (1) stipulates that each customer must be visited once. The constraint (2) guarantees that each tour starts from the depot. The constraint (3) ensures that the same vehicle arrives and leaves from each node it serves. The constraint (4) ensures that each tour ends at the depot. The constraint (5) guarantees that the service time P_i^k at any customer i by vehicle k starts inside a specified time interval $[a_i, b_i]$. The last constraint (6) prohibits the violation of the time windows. Then, if the vehicle arrives ahead of time at a customer i, it must wait until the time window $[a_i, b_i]$ opens and besides it is not allowed to arrive late.

3.3 Robust optimization

In real-world applications of operation research, we cannot ignore the fact that in the presence of uncertainties, an optimal solution could become worse or even unreachable from a practical point of view. Therefore, the need of developing models which immunize against those uncertainties has become indispensable. In this section, we present briefly the most important sets of uncertainties and the corresponding robust optimization models.

In this regard, we consider the following uncertain linear programming problem:

$$\begin{gathered} min\ c^\top x \\ s.t\ Ax \leq B \end{gathered}$$

For the remainder of this section, only the coefficients $\widetilde{a}_{ij}$ of the matrix A are object of uncertainties, and their values belong to a bounded set of uncertainties called U. Accordingly, $\widetilde{a}_{ij}$ takes value in the interval $[a_{ij} - \overline{a}_{ij}, a_{ij} + \overline{a}_{ij}]$ where a_{ij} is the nominal value and $\overline{a}_{ij}$ represents the maximum positive deviation. Therefore, we can define $\widetilde{a}_{ij}$ as:

$$\widetilde{a}_{ij} = a_{ij} + \zeta_{ij}\overline{a}_{ij}$$

Generally, ζ_{ij} is a random variable which is subject to uncertainty and varies between -1 and 1.

For instance, three types of uncertainty sets can be distinguished .

3.3.1 Box uncertainty set

The box uncertainty set is an uncertainty structure which takes the name from the box formed by the interaction of perturbations. It aims at finding a conservative solution for a robust problem where the value of all uncertain coefficients perturbs is less than a perturbation bound Ψ_i (see for e. g. Ben-Tal and Nemirovski, 1998). Its uncertainty set can be described as follows:

$$U^A = \{\widetilde{a}_{ij} = a_{ij} + \zeta_{ij}\overline{a}_{ij} \mid \ \mid \zeta_{ij} \mid \leq \Psi_i, \forall i\}$$

The robust counterpart of the problem is given by the follow:

$$\begin{aligned}
&min\ c^\top x \\
s.t \quad &\sum_j a_{ij}x_j + \Psi_i \sum_j \bar{a}_{ij}y_j \leq b_i \quad \forall i \\
&-y_j \leq x_j \leq y_j \quad \forall j \\
&y \geq 0
\end{aligned}$$

It is worth pointing that the problem becomes more conservative as the value of Ψ_i is increasing.

3.3.2 Ellipsoidal uncertainty set

The ellipsoidal uncertainty set comes to avoid over conservativeness and to limit the uncertainty space by eliminating a subset of uncertainty. The level of robustness can be controled by modifying the value of the parameter Ω wich defines the boarders of the set Ben-Tal and Nemirovski, 1999. This uncertainty set can be given as:

$$U^A = \{\tilde{a}_{ij} = a_{ij} + \zeta_{ij}\bar{a}_{ij} \mid \sum_j \zeta_{ij}^2 \leq \Omega_i^2, \forall i\}$$

The robust counterpart model is expressed in the following way:

$$\begin{aligned}
&min\ c^\top x \\
s.t \quad &\sum_j a_{ij}x_j + \sum_j \bar{a}_{ij}y_j + \Omega_i \sqrt{\sum_j \bar{a}_{ij}^2 z_{ij}^2} \leq b_i \quad \forall i \\
&-y_{ij} \leq x_j - z_{ij} \leq y_{ij} \quad \forall j \\
&y \geq 0
\end{aligned}$$

The inconvenience of this robust counterpart model lies in the generation of a convex non linear programming problem, with more computational requirement in contrast to linear models.

3.3.3 Polyhedral uncertainty set

The polyhedral uncertainty set corresponds to the most frequented case of uncertainty sets (see for e.g.), defined as the set of solutions which are protected against all situations in which at most Γ_i coefficients of the i^{th} constraints are perturbed. In this case, the robust counterpart is equivalent to a linear optimization problem.

$$U^A = \{\widetilde{a}_{ij} = a_{ij} + \zeta_{ij}\bar{a}_{ij} \mid \sum_j \mid \zeta_{ij} \mid \leq \Gamma_i, \forall i\}$$

The robust counterpart of the problem can be defined as below:

$$\begin{aligned} & min\ c^\top x \\ s.t \quad & \sum_j a_{ij}x_j + \max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = |\Gamma_i|, t_i \in J_i \setminus S_i\}} \{\sum_{j \in S_i} \bar{a}_{ij}y_j + (\Gamma_i - \mid \Gamma_i \mid)\bar{a}_{it}y_t\} \leq b_i \quad \forall i \\ & -y_j \leq x_j \leq y_j \quad \forall j \\ & y \geq 0 \end{aligned}$$

Where J_i represents the set of coefficients a_{ij} of the i^{th} constraint which are uncertain. We define for each i, a parameter Γ_i which varies in the interval $[0, \mid J_i \mid]$. The solution of this model is immunized against all cases where coefficients up to $\mid \Gamma_i \mid$ will change, and one coefficient a_{it} changes by $(\Gamma_i - \mid \Gamma_i \mid)a_{it}$ as reported by Bertsimas and Sim, 2004.

3.4 Robust approach for the VRPTW with uncertain service and travel times

Along these lines, we study the robust VRPTW including both uncertainties in travel times and service times. Our contribution to all previous works lies, first on the choice of the Adaptive Large Neighborhood Search (ALNS) metaheuristic to integrate into our approach in order to deal with robust VRPTW. Moreover, the numerical results were tested on a set of small instances based on the reference of Solomon benchmark, and large instances of Gehring Homberger's benchmark. The studied problem generated different scenarios and each scenario is performed by using the best known sampling method, Metropolis version algorithm of Monte Carlo simulation. It is worth mentioning that the ALNS approach used three different removal operators to maintain the diversity during the searching process namely: proximity operator, route portion operator and longest detour operator, and one repair operator based on the greedy insertion. For a complete description of this ALNS approach including construction and destruction operators, we refer the reader to the section 2.2.

3.4.1 ALNS applied to the robust VRPTW

In this section, we apply the adaptive large neighborhood search (ALNS) to solve the VRPTW, assuming that the displacement and the service times are both objects of uncertainty. The robustness of this approach has been tested on several scenarios generated by the Monte Carlo tool of simulation.

The proposed metaheuristic (ALNS) is an extension of the Large Neighborhood Search (LNS) heuristic, which is first introduced by Shaw, 1998, ALNS is a metaheuristic proposed by Røpke and Pisinger, 2006. It is a common technique used to enhance a locally optimal solution and can prevent getting stuck in premature convergence to local optima within tightly constrained search space. Given an initial solution obtained by a construction method, it is based on the idea of improving the initial solution by applying various destroy and repair operators to generate large neighborhoods through which the search space is explored (Pamela, Salazar-Aguilar, and Laporte, 2017). The ALNS has already been adapted to several transportation problems including vehicle routing (Røpke and Pisinger, 2006), arc routing (Salazar-Aguilar, Langevin, and Laporte, 2012), inventory-routing (Coelho, Cordeau, and Laporte, 2012), and the reliable multiple allocation hub location problem (Chaharsooghi, Momayezi, and Ghaffarinasab, 2017). The ALNS seems well-suited for the VRPTW, its power is manifested in the fact that each new solution is obtained by first removing a number of vertices, then re-inserting these vertices into the solution. ALNS was chosen because it outperforms other mono-objective algorithms applied to the same problem while keeping the simplicity and high performance that characterize local search algorithms.

We will now describe how we have adapted the general ALNS to the robust VRPTW. Our goal is to provide, for each pair of degrees of robustness (Λ, Γ) considered, a robust solution that best protects from the violation of time windows, or a solution that minimizes the total delay compared to the dates at the latest. Where Γ and Λ are two degrees of uncertainty defined to control the number of uncertain displacement times and the number of uncertain service times. They vary respectively between 0 and $\mid N \mid + \mid V \mid$, and 0 and $\mid N \mid$. Thus, when Γ=0 and Λ=0, the robust case coincides with the deterministic case, and when $\Gamma = \mid N \mid + \mid V \mid$ and $\Lambda = \mid N \mid$ is the worst case where all the displacement times and all service times are assumed uncertain and simultaneously reach their worst values.

For each pair of degrees of robustness (Λ, Γ) given, we first generate a set of possible realizations. Each realization $\omega_N^{\Lambda,\Gamma}$ is defined by the assignment of Γ displacement time (Λ service time) to their maximal values, and $\mid A \mid - \Gamma$ (resp. $\mid N \mid - \Lambda$) that remain at their nominal values. Then, on each achievement, we seek a robustly feasible solution or a solution with a minimum total delay. The reached solutions for the realizations considered are evaluated on all possible realizations and at the end of each iteration, we keep the solution that offers the worst minimum assessment. We note that the diversification of solutions is ensured by having solutions calculated independently from different realizations. Our algorithm is presented in detail in the subsections: (3.2.1), (3.2.2), (3.2.3) and (3.2.4).

Here are a few notations used in our Algorithm:

$\omega_N^{\Lambda,\Gamma}$: A realization possible

Sol_{best} : The best robust solution

Sol_N : The solution found to the N^{th} realization

$Cost(.)$: A function used to calculate the total time of displacement of a solution

$Weval^{\Gamma}(.)$: A function used to calculate the worst evaluation of a solution

$T_k = (i_1 = o, i_2, ..., i_n = o)$: The tour of the vehicle k

$p_h = (i_1, i_2, ..., i_h)$: A path of the Tour T_k

$\omega\Delta^{\Gamma,h}$: The whole of the arcs which have the Γ more large deviations of travel time

$\omega\delta^{\Lambda,h}$: The whole of the arcs which have the Λ more large deviations of service time

$\xi(p_h) = \{i_1, i_2, ..., i_h\}$: All of the Nodes which constitute p_h

$Arc(p_h) = \{arc_1 = (i_1, i_2), arc_2 = (i_2, i_3), ..., arc_{h-1} = (i_{h-1}, i_h)\}$: The whole of the arcs which constitute the path p_h

$(\overset{\_k}{s_h})$: The maximum date of arrival of the vehicle k at node i_h

Here is the detailed algorithm of our robust approach:

Algorithm 11 The robust apporach Algorithm

```
Parameters: Set $Solutions$, Set $realizations$
Outputs: Solution $solution$
$realizations \longleftarrow MonteCarlo()$
for each $realization \in realizations$ do
  $solution \longleftarrow ALNS(realization)$
  solutions.add(solution)
end for
for each $solution \in solutions$ do
  if $checkRobustness(solution) \neq True$ then
    solutions.remove(solution)
    return NULL
  end if
  if $WorstEval^{\Gamma}(solution) \neq True$ then
    solutions.remove(solution)
    return NULL
  end if
end for
$solution \longleftarrow MinObjective(solutions)$
return $solution$
```

Generation of realizations

An iteration of the Robust approach algorithm starts with the generation of a set of realizations (by Monte-Carlo Simulation), each realization $\omega_N^{\Lambda,\Gamma}$ represents a possible scenario in which the displacement times associated with a subset of arcs $\Psi \subset A$ of cardinality Γ take their maximum values $t_{ij} = t_{ij} + \Delta_{ij}$, and service times of a subset of vehicles $\theta \subset V$ of cardinality Λ take their maximum values $P_i = P_i + \delta_i$. While the other arcs and the other nodes take respectively their nominal values t_{ij} and P_i

Research of solution

For each realization $\omega_N^{\Lambda,\Gamma}$, we apply the Adaptive large neighborhood search in order to obtain a feasible solution noted Sol_N satisfies each scenario that we have already generalized by Monte-Carlo.

Check of Robustness

Even if the solution Sol_N is achievable on the realization $\omega_N^{\Lambda,\Gamma}$, it may violate window time constraints if it considers other realizations. Thus, to verify the robustness of this solution we apply Algorithm 12, where we seek to verify the robustness of the solution without the need to test it on all possible realizations. Indeed, a solution Sol_N is robustly feasible if its tours respect the windows of time at any node visited, where at most, Γ displacement times and Λ service times are uncertain.

Formally, a tour T_k is robustly feasible, if and only if, on each path p_h $\forall h \in 2, 3..., o$, the maximum arrival date $(\overset{\_k}{s_h})$ does not violate time windows to the last node. The displacement times are determined in distinguishing between two cases: the case where the degree of uncertainty Γ is greater than or equal to the number of arcs of the path p_h. In this case, we consider only the worst realization that can arise, where all the displacement times associated with the arcs of path p_h take their maximum values. On the other hand, in the case where Γ is less than the number of arcs of the path p_h, we assign to the Γ arcs of the set $\omega\Delta^{\Gamma,h}$ having the greatest deviations the maximum values, and to arcs that do not belong to this set the nominal values. The same procedure is applied to calculate the service times.

Algorithm 12 Check of the robustness

```
$feasible \leftarrow true$
**for** $k \leftarrow 1$ to $\mid V \mid$ **do**
  **for** $l \leftarrow 2$ to $\mid \xi(Tr_k) \mid$ **do**
    calculer $ArcSet^{\Gamma,l}$ and $NodeSet^{\Lambda,l-1}$
    **for** $\lambda \leftarrow 1$ to $l-1$ **do**
      **if** $l > \Gamma + 1$ and $\gamma_\lambda \notin ArcSet^{\Gamma,l}$ **then**
        $t_{\gamma_\lambda} \leftarrow t_{\gamma_\lambda}$
      **else**
        $t_{\gamma_\lambda} \leftarrow t_{\gamma_\lambda} + \Delta_{\gamma_\lambda}$
      **end if**
    **end for**
    **for** $i \leftarrow 1$ to $l-1$ **do**
      **if** $l > \Lambda$ and $c_i \notin NodeSet^{\Lambda,l-1}$ **then**
        $P_{c_i} \leftarrow P_{c_i} + \delta_{c_i}$
      **end if**
    **end for**
    $(\overset{\_k}{s_1})^{\Gamma,\Lambda} \leftarrow 0$
    **for** $i \leftarrow 2$ to $l$ **do**
      $(\overset{\_k}{s_i})^{\Gamma,\Lambda} \leftarrow max((\overset{\_k}{s_{i-1}})^{\Gamma,\Lambda} + t_{\gamma_{i-1}} + P_{c_{i-1}}, a_{c_i})$
    **end for**
    **if** $(\overset{\_k}{s_l})^{\Gamma,\Lambda} > b_{c_l}$ **then**
      feasible takes false and the algorithm ends
    **end if**
  **end for**
**end for**
```

Evaluation on the worst of case

In this step, we evaluate the solution Sol_N on the worst of possible cases, which corresponds to the realization where the displacement times associated to the Γ arcs with the worst deviations and belonging to this solution, reach simultaneously their maximum values. First, we put in descending order all the arcs of the solution according to their maximum deviations. Then, we assign to the first Γ arcs their maximum displacement time and the remaining arcs nominal displacement time. Finally, we carry out a summation of the times obtained to determine the worst-case evaluation of cases. This step is summarized in Algorithm 13.

If the solution is not robustly feasible we also calculate the total delay that it generates, and at the end of each realization, ALNS returns a robust solution (without delay) noted Sol_n, where the cost corresponds to the lowest minimum assessment, or the ALNS returns a solution that minimizes the total delay. The costs of all the solutions found on all the realizations are compared at the end of each iteration to obtain a solution with a minimal cost or a solution with a minimal delay noted Sol_i.

Algorithm 13 Evaluation on the worst of case

```
$WorstEval^{\Gamma}(S_N) \leftarrow 0$
put in descending order all the arcs of $\gamma(S_N)$ according to their maximum deviations.
**for** $i \leftarrow 1$ to $\Gamma$ **do**
    $WorstEval^{\Gamma}(S_N) \leftarrow WorstEval^{\Gamma}(S_N) + t_{\gamma_i} + \Delta_{\gamma_i}$
**end for**
**for** $i \leftarrow \Gamma + 1$ to $\gamma(S_N)$ **do**
    $WorstEval^{\Gamma}(S_N) \leftarrow WorstEval^{\Gamma}(S_N) + t_{\gamma_i}$
**end for**
**return** $WorstEval^{\Gamma}(S_N)$
```

3.4.2 Computational experiments

Since VRPTW and RVRPTW are both NP-Hard, so as to provide perfect conclusions and comparative results, we considered several kind of instances. The robust approach examined was tested on a set of small instances based on the reference of Solomon, 1987, and large instances of Gehring & Homberger's benchmark. Since the uncertainty of RVRPTW is simulated by discrete scenarios using Monte-Carlo Simulation, the uncertain travel time of each arc and the uncertain service time at each node are generated randomly between 0 and 10, with (Λ, Γ) is the degree of robustness which represents the number of service times and the number of travel times assumed to be uncertain. The used instances are noted as follow: $Gr\_\Gamma\_\Lambda\_i$, where $Gr = \{C1, C2, R1, R2, RC1, RC2\}$ corresponds to the class name of the benchmark of Solomon and Gehring & Homberger. Γ and Λ represent the number of travel times and service times supposed uncertain. $i = \{100, 200, 400, 600, 800, 1000\}$ is an index that represents the size of the instance.

The next table 3.1 shows the results obtained for small instances (100 customers) by using Cplex for the deterministic VRPTW and the results obtained by our robust approach based on ALNS that deals with the VRPTW considering that travel times and service times are both uncertain. The column "Instance" displays the notation of the instance. The column "Initial solution" presents the initial solution with which the robust approach starts. The column "best" states the best values found by the robust approach with 10 runs while the column "mean" shows the average values found by the robust approach over 10 trials. The column "Optimal" displays the optimal solution for the deterministic VRPTW calculated by Cplex.

Instance	Initial solution	Best	Mean	Optimal
R101 _ 10 _ 10 _ 100	2176.59	1918.56	1981.45	1637.7
R106 _ 10 _ 10 _ 100	1794.75	1570.49	1603.24	1234.6
R112 _ 10 _ 10 _ 100	1292.24	1135.25	1165.39	978.7
R201 _ 10 _ 10 _ 100	1751.13	1534.99	1544.19	1143.2
C101 _ 10 _ 10 _ 100	917.76	870.46	881.17	827.3
C105 _ 10 _ 10 _ 100	1016.49	872.35	906.45	827.3
C108 _ 10 _ 10 _ 100	1172.46	982.18	1016.33	827.3
C201 _ 10 _ 10 _ 100	782.39	606.44	645.1	589.1
C205 _ 10 _ 10 _ 100	805.15	662.08	713.74	586.4
C208 _ 10 _ 10 _ 100	831.3	727.28	779.43	585.8
RC101 _ 10 _ 10 _ 100	2193.94	1882.76	1994.25	1619.8
RC105 _ 10 _ 10 _ 100	1915.73	1589.8	1713.6	1513.7
RC108 _ 10 _ 10 _ 100	1710.58	1474.84	1562.7	1114.2
RC201 _ 10 _ 10 _ 100	1659.86	1463.38	1509.13	1261.8
RC202 _ 10 _ 10 _ 100	1566.17	1398.14	1448.56	1092.3
RC205 _ 10 _ 10 _ 100	1518.52	1418.02	1453.03	1154

TABLE 3.1: Performance of our robust approach versus deterministic VRPTW (CPLEX)

The next table shows the results obtained for large instances by comparing the best-known results for the deterministic VRPTW to the results found by our robust approach based on ALNS that deals with the VRPTW considering that travel times and service times are both uncertain.

Instance	Initial solution	Best	Mean	Best known
R121 _ 25 _ 25 _ 200	6610.35	5696.41	5911.84	4784.11
R141 _ 25 _ 25 _ 400	12526.98	10939.12	11082.53	10372.31
R161 _ 50 _ 50 _ 600	26176.96	24277.04	24910.6	21131.09
R181 _ 50 _ 50 _ 800	41205.13	39011.14	39643.9	36852.06
R1101 _ 100 _ 100 _ 1000	63493.03	60506.88	61702.03	53473.26
R125 _ 25 _ 25 _ 200	5156.38	4667.38	4910.81	4107.86
R145 _ 25 _ 25 _ 400	11714.54	11250	11665.48	9226.21
R165 _ 50 _ 50 _ 600	22959.22	21231.41	21802.25	19588.89
R185 _ 50 _ 50 _ 800	37521.62	36488.37	36983.03	33723.77
R1105 _ 100 _ 100 _ 1000	58238.81	56015.9	56877.48	50876.21
R1210 _ 25 _ 25 _ 200	3919.05	3676.6	3731.17	3301.18
R1410 _ 25 _ 25 _ 400	10201.39	9847.68	10121.67	8094.1
R1610 _ 50 _ 50 _ 600	21602.23	19874.36	20350.31	17748.83
R1810 _ 50 _ 50 _ 800	36258.1	34792.04	35587.88	31086.85
R11010 _ 100 _ 100 _ 1000	52326.08	50678.87	50961.69	47992.05
R221 _ 25 _ 25 _ 200	5399.58	4785.14	4879.5	4483.86
R241 _ 25 _ 25 _ 400	11836.97	10442.34	10975.86	9210.15
R261 _ 50 _ 50 _ 600	24045.81	22070.83	22702.28	18206.8
R281 _ 50 _ 50 _ 800	34167.08	32289.08	33162.16	28114.25
R2101 _ 100 _ 100 _ 1000	51402.42	48077.63	49298.56	42188.86
R225 _ 25 _ 25 _ 200	4226.4	3625.7	3871.07	3366.79
R245 _ 25 _ 25 _ 400	10150.65	8340.94	8986.2	7128.93
R265 _ 50 _ 50 _ 600	18968.26	17022.95	17790.63	15096.2
R285 _ 50 _ 50 _ 800	29807.24	28110.59	28519.41	24285.89
R2105 _ 100 _ 100 _ 1000	48290.25	46144.16	46894.91	36232.18
R2210 _ 25 _ 25 _ 200	3312.17	2913.82	3066.73	2654.97
R2410 _ 25 _ 25 _ 400	8482.81	6679.72	7785.34	5786.4
R2610 _ 50 _ 50 _ 600	15874.92	14609.57	15165.41	12253.47
R2810 _ 50 _ 50 _ 800	25210.16	23839.6	24127.11	20401.47
R21010 _ 100 _ 100 _ 1000	42490.62	40103.23	41335.65	30215.24
C121 _ 25 _ 25 _ 200	3399.45	2915.86	3029.34	2704.57
C141 _ 25 _ 25 _ 400	9205.81	8138.18	8584.48	7152.02
C161 _ 50 _ 50 _ 600	17624.24	15511.06	16672.38	14095.64
C181 _ 50 _ 50 _ 800	29445.55	28491.39	29075.07	25030.36
C1101 _ 100 _ 100 _ 1000	52251.98	51066.63	51982.04	42478.95
C125 _ 25 _ 25 _ 200	3116.86	2929.73	3058.28	2702.05
C145 _ 25 _ 25 _ 400	9992.46	8485.01	8964.53	7152.02
C165 _ 50 _ 50 _ 600	18388.46	17227.57	18035.76	14085.72
C185 _ 50 _ 50 _ 800	30275.37	28646.79	29428.55	25166.28
C1105 _ 100 _ 100 _ 1000	52938.75	52150.1	52467.93	42469.18
C1210 _ 25 _ 25 _ 200	3148.39	2808.41	2990.42	2643.51
C1410 _ 25 _ 25 _ 400	9195.54	8208.69	8632.71	6860.63
C1610 _ 50 _ 50 _ 600	16565.12	15842.58	16210.46	13637.34
C1810 _ 50 _ 50 _ 800	29738.8	27501.34	27112.31	24070.17
C11010 _ 100 _ 100 _ 1000	50163.43	46321.77	47806.49	39858.64

TABLE 3.2: Performance of our robust approach versus best known results

Instance	Initial solution	Best	Mean	Best known
C221 _ 25 _ 25 _ 200	2316.76	2190.94	2244.38	1931.44
C241 _ 25 _ 25 _ 400	5043.61	4632.06	4957.28	4116.05
C261 _ 50 _ 50 _ 600	11601.77	10617.6	11308.25	7774.1
C281 _ 50 _ 50 _ 800	15126.33	14213.43	14833.2	11654.81
C2101 _ 100 _ 100 _ 1000	21742.75	20735.79	21212.43	16879.24
C225 _ 25 _ 25 _ 200	2444.25	2164.43	2243.59	1878.85
C245 _ 25 _ 25 _ 400	5243.39	4165.92	4405.26	3938.69
C265 _ 50 _ 50 _ 600	11527.53	10346.3	10596.96	7575.2
C285 _ 50 _ 50 _ 800	16054.91	14992.32	15575.45	11425.23
C2105 _ 100 _ 100 _ 1000	25179.55	24255.43	24799.67	16561.29
C2210 _ 25 _ 25 _ 200	2282.97	2008.49	2101.45	1806.58
C2410 _ 25 _ 25 _ 400	5012.63	4399.61	4687.95	3827.15
C2610 _ 50 _ 50 _ 600	11334.43	10294.91	10523.94	7255.69
C2810 _ 50 _ 50 _ 800	14967.74	14043.06	14651.07	10977.36
C21010 _ 100 _ 100 _ 1000	25907.63	24024.33	24624.61	15943.34
RC121 _ 25 _ 25 _ 200	4360.55	3815.01	4173.33	3602.8
RC141 _ 25 _ 25 _ 400	12220.71	10673.61	11061.17	8573.96
RC161 _ 50 _ 50 _ 600	23620.95	21917.12	22561.48	17014.17
RC181 _ 50 _ 50 _ 800	37849.35	35821.46	36257.21	31117.04
RC1101 _ 100 _ 100 _ 1000	56325.86	53322.91	54165.08	46138.01
RC125 _ 25 _ 25 _ 200	4182.22	3601.92	3762.9	3371
RC145 _ 25 _ 25 _ 400	11602.21	10604.17	10930.51	8172.64
RC165 _ 50 _ 50 _ 600	21047.61	20073.28	20400.21	16566.24
RC185 _ 50 _ 50 _ 800	35558.2	34959.07	35397.32	29796.67
RC1105 _ 100 _ 100 _ 1000	56705.85	53937.4	54407.58	45313.38
RC1210 _ 25 _ 25 _ 200	3902.19	3349.59	3514.39	3000.3
RC1410 _ 25 _ 25 _ 400	8908.94	8007.56	8397.93	7596.04
RC1610 _ 50 _ 50 _ 600	18946.53	18149.09	18591.05	15675.99
RC1810 _ 50 _ 50 _ 800	32010.09	31610.09	31838.24	28474.35
RC11010 _ 100 _ 100 _ 1000	49491.71	47296.55	48010.76	43679.61
RC221 _ 25 _ 25 _ 200	3865.62	3357.12	3460.46	3099.53
RC241 _ 25 _ 25 _ 400	9326.5	7160.2	7667.79	6682.37
RC261 _ 50 _ 50 _ 600	17021.84	16167.74	16660.9	13324.93
RC281 _ 50 _ 50 _ 800	29056.9	28307.07	28849.06	20981.14
RC2101 _ 100 _ 100 _ 1000	44947.06	43853.65	44307.34	30278.5
RC225 _ 25 _ 25 _ 200	3511.92	3232.09	3457.93	2911.46
RC245 _ 25 _ 25 _ 400	9809.92	7445.5	8708.31	6710.12
RC265 _ 50 _ 50 _ 600	17329.58	16076.16	16844.96	13000.84
RC285 _ 50 _ 50 _ 800	26883.72	26099.31	26607.32	19136.03
RC2105 _ 100 _ 100 _ 1000	42576.12	40999.51	41311.28	27140.77
RC2210 _ 25 _ 25 _ 200	2773.13	2397.68	2506.15	2015.6
RC2410 _ 25 _ 25 _ 400	5213.86	4722.23	4911.98	4278.61
RC2610 _ 50 _ 50 _ 600	12878.34	11991.88	12302.9	9069.41
RC2810 _ 50 _ 50 _ 800	20326.42	18361.79	18830.3	14439.14
RC21010 _ 100 _ 100 _ 1000	32588.02	30015.61	30598.91	21910.33

TABLE 3.3: Performance of our robust approach versus best known results

In order to visualize the impact of increasing degrees of uncertainty on objective function. We set the value of Γ to 25 and we adjusted the value of Λ for multiple instances of size 100. Here is the curve obtained:

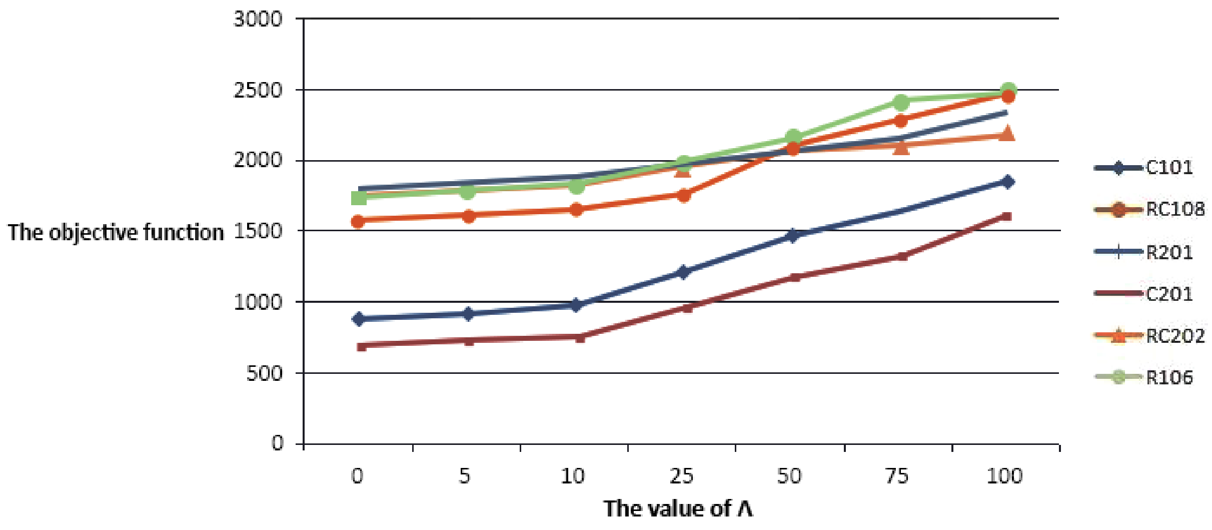

FIGURE 3.1: Objective function versus the value of Λ

The graph shows clearly that the objective function increases according to degree of uncertainty. To the best of our knowledge, this contribution is the first work to be devoted to the study of VRPTW considering both the uncertainties on travel times and service times. Due to the lack of work in this direction, we compared our results with the deterministic VRPTW literature even if the two problems are different in the sense that a good solution found for the deterministic case could become worse in the presence of uncertainties or even unreachable.

In fact, the robustness of this approach has been tested on several data sets and the results showed that the robust solutions offer great protection against delays with a slight increase in travel times and service times compared to what would have been found if a deterministic solution had been applied. The developed algorithm offers decision-making tool that allows them to choose, according to their specifications, the level of protection as well as the solution to apply. It is then clear that our approach is very powerful in terms of the robustness since it included several algorithms (Robustness verification, worst-case evaluation ...) which leads to near best solutions for all the possible realizations of the uncertainties without any further considerations but only nominal values and deviations possible uncertain data are sufficient.

3.5 Multi-threading parallel Robust approach for the VRPTW with uncertain service and travel times

The goal of this section is to study the effect of multi-threading parallelization of the resolution approach blocks on the running time and the objective function. The optimization problem considered here is a variant of the Robust Vehicle Routing Problem with Time Windows (RVRPTW) including both uncertainties in travel and service times. Our contribution to all previous works lies first on the choice of the efficient Parallel Adaptive Large Neighborhood Search (PALNS) metaheuristic of Røpke, 2009, which leads to a reduced running time. Moreover, we used our parallel version of the Metropolis Monte Carlo algorithm to generate all possible realizations and to transform the problem under uncertainties to a set of deterministic sub-problems. For more detail about this splitting process, see for instance the previous work Nasri et al., 2020a and the references therein. Based on the efficient implementation of Røpke, 2009, different combinations (sequential/parallel) of the Monte Carlo algorithm and ALNS are performed. In this way, our strategy offers to decision makers the choice of the combination depending on their preferences and the situation at hand.

To the best of our knowledge, this contribution is the first work to be devoted to the study of VRPTW considering the uncertainties on travel times and service times for different sizes of instances in terms of both the execution time and the objective value.

3.5.1 Motivation of our parallel robust approach

In this part, we will provide a detailed exposition of the multi-threading parallel approach. We start by giving some insights of the motivation before to handle the complete description of the proposed approach.

The first challenge of such approach is to derive the best robust solution that responds to to all uncertainties with reduced running time. However, the sequential robust approach suffers from lengthy computational times; partly because the generation of scenarios is time-consuming as well as the research of solution block. In the contrast of the check of robustness and the evaluation on the worst case blocks which have minimal running time, since they do not contribute to solve the problem, but they deal with evaluating the obtained solution.

Instance size	Monte-Carlo block/iteration	ALNS block/iteration	Check of robustness block/iteration	Worst-Case evaluation block/iteration
1000 _ 100 _ 100	210	258	46	30
1500 _ 100 _ 100	325	409	85	52
2000 _ 100 _ 100	478	632	217	96
2500 _ 100 _ 100	621	867	377	163
3000 _ 100 _ 100	1042	1436	510	294
3500 _ 100 _ 100	1893	2229	682	325
4000 _ 100 _ 100	2365	3538	793	549
4500 _ 100 _ 100	2901	4428	907	703

TABLE 3.4: The execution time of the sequential robust approach blocks in (ms)

Table 3.4 confirms our assertion that the generation of scenarios and the ALNS blocks take considerable time compared to the other blocks. In order to overcome this impediment, we propose a multithreading parallelization of the costly blocks as detailed in the next subsections.

3.5.2 The parallel Monte Carlo sampling

The parallel Metropolis Monte Carlo algorithm described below is a scenario generation technique that uses a defined number of worker threads to generate in a parallel way a predefined number n of independent identically distributed scenarios. In practice, each worker thread produces a realization $R_l^{\Lambda,\Gamma}$ that corresponds to a deterministic VRPTW problem, in which Γ displacement times and Λ service times reach simultaneously their maximum values. The pseudocodes of this parallel method is shown in the algorithm:

Algorithm 14 Parallel Monte-Carlo

```
Input: $\Lambda, \Gamma, n$
Output: scenarios
for $l \leftarrow 1$ to $n$ in parallel do
    for $k \leftarrow 1$ to $\Gamma$ do
        Select randomly two clients $i$ and $j$
        $\widetilde{t_{ij}} \leftarrow t_{ij} + \Delta_{ij}$
    end for
    for $k \leftarrow 1$ to $\Lambda$ do
        Select randomly a client $i$
        $\widetilde{P_i} \leftarrow P_i + \delta_i$
    end for
    Generate a scenario $R_l^{\Lambda,\Gamma}$
    $t(R_l^{\Lambda,\Gamma}) \leftarrow \widetilde{t}(R_l^{\Lambda,\Gamma})$
    $P(R_l^{\Lambda,\Gamma}) \leftarrow \widetilde{P}(R_l^{\Lambda,\Gamma})$
    Add $R_l^{\Lambda,\Gamma}$ to $scenarios$
end for
return $scenarios$
```

3.5.3 The parallel ALNS

In this subsection, we present the Parallel Adaptive Large Neighborhood Search (PALNS) method developed by Røpke, 2009. This method can be presented in three phases (see Figure 3.2).

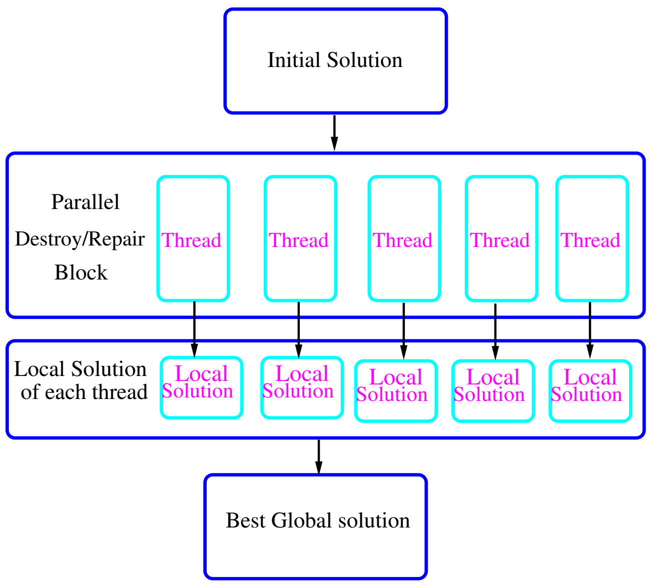

FIGURE 3.2: The Parallel Adaptive Large Neighborhood Search

At the first level, we generate an initial feasible solution by using the greedy insertion metaheuristic Solomon, 1987. The main idea behind this method is to select the best feasible insertion place in the incumbent route for each non-inserted node taking into account two major factors: the increase in total cost of the current route after the insertion and the delay of the service start time of the customer succeeding the newly inserted customer.

The second phase is related to a set of destroy and repair operators designed to enhance the incumbent solution. In this context, each worker thread deals with a copy of the current solution and executes destroy and repair methods on this local copy in order to improve it.

The third phase collects the routes of different local solutions that each thread has obtained, for the purpose of combining it into a new better temporary global solution and sending it to be improved. At this stage, we will accept or reject the generated solution, based on a hill climbing acceptance criterion. It is worth mentioning that only the current and global best solutions are shared between worker threads, in order to update them as necessary by repeating the process until a stop criterion is met.

We should point out here that the ALNS uses a flexible layer with a set of destruction heuristics (proximity operator, route portion operator, and longest detour operator) and an insertion heuristic (the greedy insertion) and applies them by a roulette wheel selection that highlights the corresponding performance obtained during the search. On the other hand, the LNS heuristic does not use this scoring mechanism.

For a completed description of the used PALNS, we refer the reader to the study of Røpke, 2009 and the references therein.

3.5.4 Computational experiments

In this section, we summarize a few of the results obtained by evaluating different robust approaches conceived of to solve the vehicle routing problem with time windows with uncertain service and travel times.

We explore the effect of applying the thread parallelism to the Monte Carlo and ALNS blocks, on the execution time and the objective value. This leads to four different robust combinations: the sequential approach that uses sequential Monte Carlo and sequential ALNS (MC sequential and ALNS sequential), the approach employing sequential Monte Carlo and parallel ALNS (MC sequential and ALNS parallel), the approach that uses parallel Monte Carlo and sequential ALNS (MC parallel and ALNS sequential), and the parallel approach combining parallel Monte Carlo and parallel ALNS (MC parallel and ALNS parallel). We should note here that the sequential approach (MC sequential and ALNS sequential) coincides with the only method from the literature Nasri et al., 2020a that deals with the considered problem.

It is important to mention that the notion of thread parallelism used in our context can be defined as the capability of a processing unit to execute multiple processes contemporaneously or with time slicing. By means of a thread, the smallest unit of processing can be performed in an operating system in order to accelerate the execution time and manage the code over time. In our study, we use four threads in order to establish the comparison between the different approaches. The choice of four threads is not restrictive, and we can use as many threads as possible. Our assumption is that using more threads leads to the improvement of the average execution time, but it slightly decreases the quality of the obtained solution. For more details, see, e.g., Røpke, 2009.

The robust approaches studied in this dissertation were tested on a classical set of instances in reference to Solomon's benchmark (1987) Solomon, 1987 and Gehring and Homberger's benchmark Gehring and Homberger, 1999:

- Set R contains problems with randomized customers.
- Set C contains problems with clustered customers.

- Set *RC* contains problems with both clustered and randomized customers.

In order to simulate the uncertainty of RVRPTW by discrete scenarios, the uncertain travel time and uncertain service time are generated at random, so that they go from 0 to 10. We denote the used instances as follows: Gr_Γ_Λ, where Γ and Λ present respectively the number of travel times and service times considered uncertain and *Gr* refers to the instance of Solomon and Gehring and Homberger's benchmark or the size of larger instances.

For small instances (Solomon and Homberger's instances), we chose 15,600 iterations as a stop criterion in order to diversify the research, which may ameliorate the quality of our solution, because the solution has a greater chance to escape from a local minimum. As far as we are aware, the maximal number of iterations for instances is falling in the literature. With a view toward examining the capability of our approaches for tackling that problem, we judge it based on the average performance over 10 multiple independent runs.

For the set of instances larger than 1000, we generate random representative instances in such a manner that the travel time between each pair of nodes is between 0 and 100, and the same for the service time. The time interval has a capacity of 200 between the start and the end of the service at each customer. We forced a stop condition of about 20 min, which allows a good comparison between the proposed methods in terms of the number of reached iterations for the same time interval. Then, we present the measurements achieved for a single run.

The proposed algorithms were implemented in Java 7, compiled with Intel compiler Celeron 1.80 GHz core i5 with 8 GB RAM.

Execution Time

Table 3.5 presents a comparison of the execution time for each instance group between different robust approaches with the maximal number of iterations of 15,600. As expected, the results show that the approaches containing the parallel ALNS succeeded by those containing parallel Monte Carlo lead to the improvement of the average execution time compared to other sequential approaches. This can be explained by the fact that the ALNS block succeeded by the Monte Carlo block consumes most of the execution time compared to other blocks.

In the same spirit, we report in Table 3.6 a comparison of different approaches according to the number of reached iterations when the stopping limit time is about

20 min for the group of instances 2500–4500. The approach containing more parallel blocks attained more iterations for the same time interval since it reduced the execution time of the consuming blocks.

Table 3.7 depicts the improvement results in execution time for Solomon's instances. The conclusion from this table is clear: the running time is much faster for the approaches using parallel ALNS succeeded by those employing the parallel Monte Carlo algorithm.

Instance	**MC Sequential and ALNS Sequential**	**MC Sequential and ALNS Parallel**	**MC Parallel and ALNS Sequential**	**MC Parallel and ALNS Parallel**
100 _ 10 _ 10	832.24	701.03	725.45	574.24
200 _ 25 _ 25	1007.41	853.11	883.25	669.33
400 _ 25 _ 25	1331.47	1037.98	1101.12	770.91
600 _ 50 _ 50	1402.50	1160.24	1230.64	858.38
800 _ 50 _ 50	1545.39	1302.85	1331.68	979.14
1000 _ 100 _ 100	2687.50	2062.25	2165.74	1340.49
1500 _ 100 _ 100	4519.79	3267.98	3665.37	2413.56
2000 _ 100 _ 100	5606.81	3953.18	4389.74	2736.11

TABLE 3.5: The execution time of different robust approaches in seconds for 15,600 iterations.

Instance	**MC Sequential and ALNS Sequential**	**MC Sequential and ALNS Parallel**	**MC Parallel and ALNS Sequential**	**MC Parallel and ALNS Parallel**
2500 _ 100 _ 100	3038	4275	3879	5681
3000 _ 100 _ 100	2163	3572	2794	4141
3500 _ 100 _ 100	1375	2992	1985	3709
4000 _ 100 _ 100	912	2401	1386	3124
4500 _ 100 _ 100	594	1994	1078	2300

TABLE 3.6: Number of iterations reached in 20 min.

Instance	**MC Sequential and ALNS Sequential**	**MC Sequential and ALNS Parallel**	**MC Parallel and ALNS Sequential**	**MC Paralle and ALNS**
R101 _ 10 _ 10	433	267	378	212
C101 _ 10 _ 10	491	320	434	263
RC101 _ 10 _ 10	468	286	407	225
R201 _ 10 _ 10	487	301	425	239
C201 _ 10 _ 10	563	367	497	301
RC201 _ 10 _ 10	553	338	482	267
R121 _ 25 _ 25	580	277	512	309
C121 _ 25 _ 25	653	442	583	372
RC121 _ 25 _ 25	605	406	539	340
R221 _ 25 _ 25	673	438	595	360
C221 _ 25 _ 25	784	531	700	447
RC221 _ 25 _ 25	707	475	629	397
R141 _ 25 _ 25	668	423	586	341
C141 _ 25 _ 25	775	538	696	459
RC141 _ 25 _ 25	747	504	666	423
R241 _ 25 _ 25	937	629	823	479
C241 _ 25 _ 25	981	683	877	589
RC241 _ 25 _ 25	957	646	854	543
R161 _ 50 _ 50	741	541	675	475
C161 _ 50 _ 50	907	682	832	607
RC161 _ 50 _ 50	825	607	753	535
R261 _ 50 _ 50	923	730	909	640
C261 _ 50 _ 50	1089	814	999	726
RC261 _ 50 _ 50	1051	774	958	681
R181 _ 50 _ 50	923	698	848	623
C181 _ 50 _ 50	942	732	874	664
RC181 _ 50 _ 50	929	706	855	632
R281 _ 50 _ 50	1136	860	1044	768
C281 _ 50 _ 50	1249	971	1157	879
RC281 _ 50 _ 50	1223	925	1123	829
R1101 _ 100 _ 100	1179	764	1040	625
C1101 _ 100 _ 100	1213	817	1081	685
RC1101 _ 100 _ 100	1204	800	1069	665
R2101 _ 100 _ 100	1757	1140	1552	935
C2101 _ 100 _ 100	1815	1224	1618	1027
RC2101 _ 100 _ 100	1775	1181	1577	983

TABLE 3.7: Solomon's instance: Comparison of the runtimes in seconds of different robust approaches.

Objective Function

Table 3.8 presents the objective function of different robust approaches for the group of instances 2500-4500. When the size of the instance increases, the cost function of the approaches containing parallel and sequential ALNS solution converges. Then, we compute the mean absolute percent deviation (MAPD), which is the absolute difference between the cost function of the approach containing the sequential ALNS and the parallel ALNS divided by the magnitude of the objective function in the approach with sequential ALNS. This indicator (MAPD) goes from 13.05% for the instance of size 1000 to 3.96% for the instance of size 4500. We can conclude that incorporating the parallel ALNS in the approaches is efficient for large instances.

Table 3.9 depicts the results of the objective value for some of Solomon's instances. We observe that the ALNS controls the solution quality. Then, the approaches that contain a sequential ALNS yield better results than those that contain parallel ALNS. The ALNS is responsible for finding the solution of each scenario, in contrast with the Monte Carlo algorithm, which is limited to generating the possible scenarios. When we increase the instance size, the quality of the solution of the parallel approach becomes more interesting.

Instance	**MC Sequential and ALNS Sequential**	**MC Sequential and ALNS Parallel**	**MC Parallel and ALNS Sequential**	**MC Parallel and ALNS Parallel**
1000 _ 100 _ 100	18,612	21,031	18,642	21,075
1500 _ 100 _ 100	25,961	28,557	25,904	28,516
2000 _ 100 _ 100	36,420	38,969	36,408	38,943
2500 _ 100 _ 100	44,981	48,129	44,998	48,161
3000 _ 100 _ 100	52,817	55,986	52,836	56,014
3500 _ 100 _ 100	60,356	63,977	60,321	63,937
4000 _ 100 _ 100	67,675	71,059	67,642	70,986
4500 _ 100 _ 100	75,306	78,318	75,329	78,371

TABLE 3.8: Comparison of different approaches according to the objective function

Instance	**MC Sequential and ALNS Sequential**	**MC Sequential and ALNS Parallel**	**MC Parallel and ALNS Sequential**	**MC Paralle and ALNS**
R101 _ 10 _ 10	1918.56	2225.52	1926.14	2249.69
C101 _ 10 _ 10	870.46	1009.73	883.01	1025.84
RC101 _ 10 _ 10	1882.76	2145.48	1897.04	2183.92
R201 _ 10 _ 10	1434.99	1663.44	1415.09	1663.24
C201 _ 10 _ 10	666.54	779.27	663.52	761.51
RC201 _ 10 _ 10	1663.38	1895.82	1679.45	1900.11
R121 _ 25 _ 25	5696.41	6379.52	5709.64	6659.85
C121 _ 25 _ 25	3115.86	3519.95	3115.86	3504.38
RC121 _ 25 _ 25	3915.01	4384.80	3919.63	3492.62
R221 _ 25 _ 25	5385.14	6031.35	5385.14	6024.07
C221 _ 25 _ 25	2490.94	2813.70	2496.91	2813.70
RC221 _ 25 _ 25	3657.12	4095.84	3657.12	4106.49
R141 _ 25 _ 25	10,939.12	11,485.95	10,951.24	11,485.95
C141 _ 25 _ 25	8138.18	8463.52	8127.63	8480.31
RC141 _ 25 _ 25	10,673.61	11,099.92	10,697.27	11,080.79
R241 _ 25 _ 25	10,442.34	10,859.68	10,463.29	10,843.81
C241 _ 25 _ 25	4932.06	5129.28	4863.65	5129.28
RC241 _ 25 _ 25	20,917.12	21,690.92	7181.07	21,709.56
R161 _ 50 _ 50	24,277.04	24,714.02	24,277.04	24,703.48
C161 _ 50 _ 50	15,511.06	15,945.30	15,523.45	15,940.06
RC161 _ 50 _ 50	21,917.15	22,245.75	21,937.13	22,291.10
R261 _ 50 _ 50	22,070.80	22,445.19	22,121.72	22,457.14
C261 _ 50 _ 50	10,617.60	10,871.80	10,632.96	10,886.71
RC261 _ 50 _ 50	16,167.74	16,458.00	16,186.14	16,430.92
R181 _ 50 _ 50	39,011.14	39,479.13	38,979.56	39,431.41
C181 _ 50 _ 50	28,491.39	29,117.85	28,491.39	29,117.83
RC181 _ 50 _ 50	35,821.46	36,358.31	35,853.12	36,347.02
R281 _ 50 _ 50	32,289.08	32,870.20	32,271.22	32,870.20
C281 _ 50 _ 50	14,213.43	14,483.04	14,219.34	14,497.15
RC281 _ 50 _ 50	28,307.07	28,788.21	28,307.07	28,805.09
R1101 _ 100 _ 100	60,506.88	62,805.22	60,493.16	62,805.22
C1101 _ 100 _ 100	52,251.98	53,766.83	52,257.48	53,742.30
RC1101 _ 100 _ 100	53,322.91	55,188.27	43,318.10	55,188.27
R2101 _ 100 _ 100	48,103.23	49,449.88	48,103.23	48,463.23
C2101 _ 100 _ 100	20,735.79	21,357.05	20,700.14	21,357.05
RC2101 _ 100 _ 100	43,853.65	44,905.47	43,853.65	44,884.35

TABLE 3.9: Solomon's instance: comparison of the objective value of different robust approaches.

3.6 Conclusion

Our main goal in this chapter was to consider the robust vehicle routing problem with time windows under both travel times and service times uncertainties. For this purpose, a new robust approach has been suggested to minimize the total distance of the travel time in the presence of the maximum deviations of possible uncertain data. In this contrast, we generate all possible scenarios by using Monte Carlo simulation and we opt for the adaptive large neighborhood search ALNS algorithm to solve each sub-problem related to each scenario. In this context, several destroy/repair method is combined to explore multiple neighborhoods within the same search and defined implicitly the large neighborhood. In order to study the feasibility of the resulting solution, efficient mechanisms have been conceived, the first concerns the verification of the robustness, while the second takes into consideration the evaluation of the solution on the worst case.

This new approach lying in the introduction of an effective way of modeling and handling several uncertainty data levels defined by pairs of uncertainty (Λ, Γ), which represent respect-ively the number of service times and the number of travel times assumed uncertain, has been tested on several sets of problems and showed improved robustness results for bench-mark instances. The computational experiments were performed to examine our proposed new approach compared to the deterministic VRPTW literature on a set of small instances based on the Solomon VRPTW benchmark and large instances of Gehring & Homberger benchmark.

As far as the adopted approach derives the best robust solution that responds to all uncertainties, it still suffer from lengthy computational times; partly because the generation of scenarios, as well as the research of the solution block are time-consuming. As an alternative to remedy this problem, we introduce a procedure for the thread parallelism in the Monte Carlo block, and we use the parallel ALNS proposed by Røpke, 2009. This leads to four different robust approaches combining the (sequential/parallel) Monte Carlo algorithm and the (sequential/parallel) ALNS.

The considered approaches are tested on Solomon's benchmark instance of VRPTW and lager instances generated randomly. Accordingly, we can offer a decision-making solution that provides great protection against delays in a reasonable running time. However, we should note that the related counterpart of using the parallel ALNS, which is the objective value, can be influenced, since the parallel ALNS slightly reduces the quality of the solution, especially for small instances.

Future work will focus on the integration of a pre-processing step based on different clustering techniques such as K-means, K-medoids, density-based spatial, etc., in order to ensure the commitment of the solutions. These techniques will not change the structure of the suggested approach drastically, and our assumption is that they will enhance the solution quality obtained with the parallel ALNS.

Chapter 4

Conclusion

In this dissertation, we explored the idea of developing efficient methods to solve two main problems that model many real world issues in the field of the vehicle routing problems. The solution proposed approaches include both sequential and parallel methods. To evaluate these methods, numerical experiments and comparisons using different instance sizes were provided.

After presenting the studied problem and giving some insights of motivation behind our work in the first chapter, the dissertation included two chapters where each, focused on one of the variants of the vehicle routing problem.

The first chapter investigated a well known problem referred as the Vehicle Routing Problem with Time Windows (VRPTW) and included four sections. In the first section, we exposed the Adaptive Large Neighborhood Search metaheuristic, which seems well-suited for the VRPTW, its power is manifested in the fact that each new solution is obtained by first removing a number of vertices, then re-inserting these vertices into the solution. ALNS was chosen because it outperforms other mono-objective algorithms applied to the same problem while keeping the simplicity and high performance that characterize local search algorithms.

The second section presented different approaches conceived to enhance the process of the ALNS in terms of objective function. The first approach consists on incorporating the modified choice function (MCF) (Drake, Özcan, and Burke, 2012) into the selection phase of the ALNS while sparing the whole process. In other words, the destroy and repair operators will not be selected by the roulette wheel mechanism as in the original ALNS, but they will be rather selected by the MCF. We also took advantage of clustering research from other domains to revisit clustering approaches to routing. We proposed a hierarchical approach consisting of two stages as Cluster first - Route second. Our solution technique is based on creating a set of cost-effective feasible clusters, that form certain VRPTW sub-problems with respect to the depot, using k-medoid algorithm within an effective spatio-tamporal distance similarity which is totally appropriate to the nature of the VRPTW. Therefore, we obtained a routing solution by ordering all customers in every cluster separately using three distinct routing algorithms (i.e., ALNS, GA and VNS).

The third section provided a new approach of parallel adaptive large neighborhood search algorithm designed to solve the vehicle routing problem with time windows. The objective here was to select a "good" solution in a reasonable time without too much compromising the quality of the solution. The major process introduced a technique for the Thread parallelism of greedy insertion used in the initial block and also the destroy/repair operators involved in the mechanism of the ALNS. As an improvement of this approach, we used an iterative clustering algorithm as preprocessing step to perform initial data and to affect them to the worker thread. We used K-means for the clustering algorithm as prototype which it is not restrictive method but of course, we can adopt other techniques such as K-medoids or density based spatial clustering. We showed how the integration of thread parallelism applied to destruction/repair heuristics improved the execution time. It reached competitive results especially for the large instances of Solomon and Homberger.

In the second part of the dissertation, we focused on a new variant of VRPTW incorporating service time as a source of uncertainty in addition to the travel times. We considered a sequential robust approach which requires efficient mechanisms to derive a best robust solution that responds to all uncertainties. The problem is called then, the Robust Vehicle Routing Problem with Time Windows (RVRPTW). After using this sequential approach which include generating all possible scenarios by using Monte Carlo simulation and using the adaptive large neighborhood search ALNS algorithm to solve each sub-problem related to each scenario, a new thread parallelism strategy was introduced in those blocks. This leads to four different robust approaches combining the (sequential/parallel) Monte Carlo algorithm and the (sequential/parallel) ALNS. The experiments conducted on Solomon's benchmark and larger generated instances clearly demonstrated the efficiency and the competitiveness of the robust approach especially when considering large instances.

The primary goal of this thesis was the study of new solution approaches for the vehicle routing problem with time windows with or without uncertainties. The observations made in these works present some interesting open questions. In the following, we outline some directions for future research:

- We plan to include a pre-processing step based on different clustering techniques such as K-means, K-medoids, density-based spatial, etc., in order to ensure the commitment of the solutions obtained for the robust problem. These techniques will not change the structure of the suggested approach drastically, and our assumption is that they will enhance the solution quality obtained with the parallel ALNS.
- We intend to study in depth the the parameters settings and the performance of the algorithm notably when adopting different linkage metrics to measure the proximity between clusters.
- The proposition of a multi-objective algorithm which deals with the objective value and the execution time would be an interesting area, dealing with both of the studied problem.

Appendix A

Additional results & Algorithms

.1 Figures

The execution time consumed per iteration of the suggested thread parallelism approach depends on the level of computing: initial phase, repair and destroy stages. Therefore, in this section we compared both ALNS approaches: sequential and parallel.

Figures 3-6 illustrated the execution time versus the instance size of repair operator: parallel greedy insertion (see fig. 1) and the three different destruction operators: proximity (see fig. 2), route portion (see fig. 3), longest detour (see fig. 4).

It is clear from different figures that the sequential approach consumes more run-time than our multi-threading parallel approach. The speedup factor becomes more significant as instance size is increasing in particular for instances above 1000. So whenever multiple threads are available, it is expected to enhance the parallelization to get better solutions in a shorter time. Indeed, the use of cheapest clustering algorithm (K-means, K-medoïd, . . .) in terms of run-time may improve the quality of our solution and increase the objective function.

FIGURE 1: The execution time of sequential and parallel greedy insertion operator per iteration in (ms)

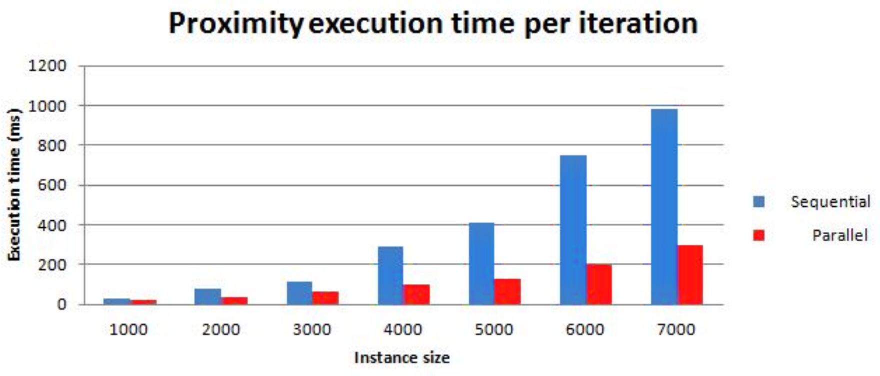

FIGURE 2: The execution time of sequential and parallel proximity operator per iteration in (ms)

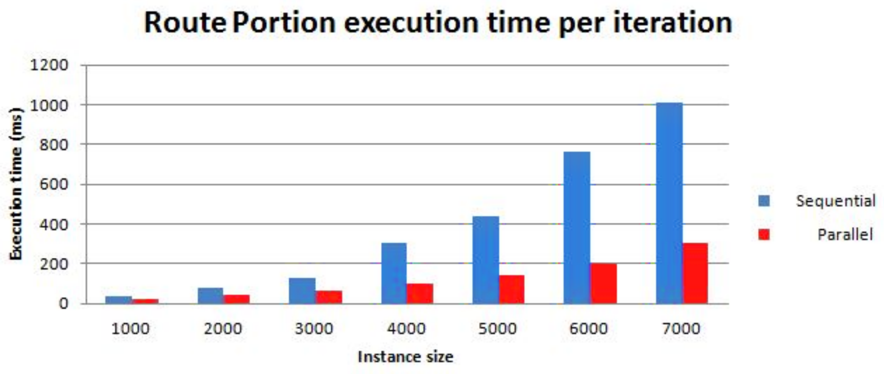

FIGURE 3: The execution time of sequential and parallel portion-remove operator per iteration in (ms)

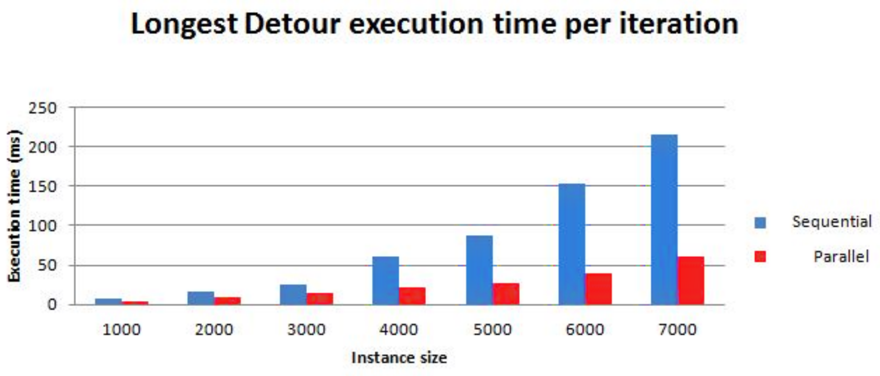

FIGURE 4: The execution time of sequential and parallel longest detour operator per iteration in (ms)

.2 Algorithm: find closest

As mentioned before, the parallel greedy insertion algorithm uses the find closest function. The best closed location of a given node is done by constructing a new route for each thread and the nearest node is found by the function findClosest in line 8 of the algorithm 2. The detail of this function is described in the algorithm below.

Algorithm 15 findClosest function

```
1: Inputs: Set $remainingNodes$
2: Outputs: node $bestNode$
3: $bestVal = MaxValue$
4: for $node \in remainingNodes$ do
5:   $c_1(pred(node), node, succ(node)) = \alpha_1 c_{11}(pred(node), node, succ(node)) + \alpha_2 c_{12}(pred(node), node, succ(node))$
6:   if $c_1(pred(node), node, succ(node)) < bestVal$ then
7:     $bestVal = c_1(pred(node), node, succ(node))$
8:     $bestNode = node$
9:   end if
10: end for
11: return $bestNode$
```

Bibliography

Afifi, S., D. C. Dang, and A. Moukrim (2013). "A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints". In: *International Conference on Learning and Intelligent Optimization: Lecture Notes in Computer Science* 7997.

Agra, A. et al. (2012). "Layered Formulation for the robust vehicle routing problem with time windows". In: *Lecture Notes in Computer Science* 7422, pp. 249–260.

— (2013). "The robust vehicle routing problem with time windows". In: *Computers and Operations Research* 40, pp. 856–866.

Alumur, S., S. Nickel, and F. Saldanha da Gama (2012). "Hub location under uncertainty". In: *Transportation Research Part B: Methodological* 46, pp. 529–543.

Azi, N., M. Gendreau, and J. Y. Potvin (2010). "An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles". In: *European Journal of Operational Research* 202, pp. 756–763.

Baker, E. and J. Schaffer (1986). "Computational experience with branch exchange heuris-tics for vehicle routing problems with time window constraints". In: *American Journal of Mathematical and Management Sciences* 6, pp. 261–300.

Baron, O., J. Milner, and H. Naseraldin (2011). "Facility location: A robust optimization approach". In: *Production and Operations Management* 20, pp. 772–785.

Ben-Tal, A., G. Boaz, and S. Shimrit (2009). "Robust multi-echelon multi-period inventory control". In: *European Journal of Operational Research* 199, pp. 922–935.

Ben-Tal, A. and A. Nemirovski (1998). "Robust convex optimization". In: *Operations Research Letters* 23, pp. 769–805.

— (1999). "Robust solutions of uncertain linear programs". In: *Operations Research Letters* 25, pp. 769–805.

Bertsimas, D. and M. Sim (2003). "Robust discrete optimization and network flows". In: *Mathematical programming* 98, pp. 49–71.

— (2004). "The price of robustness". In: *Operations Research* 52, pp. 35–53.

Bienstock, D. and N. Özbay (2008). "Computing robust basestock levels". In: *Discrete Optimization* 5, pp. 389–414.

Bouthillier, A. and T. G. Crainic (2005). "A cooperative parallel meta-heuristic for the vehicle routing problem with time windows". In: *Computers & Operations Research* 32, pp. 1685–1708.

Braekers, K., K. Ramaekers, and I. Nieuwenhuyse (2015). "The Vehicle Routing Problem: State of the Art Classification and Review". In: *Computers & Industrial Engineering* 99.

Bräysy, O. and M. Gendreau (2002). "Tabu Search heuristics for the Vehicle Routing Problem with Time Windows". In: *Top Journal* 10, 211—237.

— (2005a). "Vehicle Routing Problem with time windows, Part II: Metaheuristics". In: *Journal of Transportation Science* 39, pp. 119–139.

— (2005b). "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms". In: *Journal of Transportation Science* 39, pp. 104–118.

C. Pinar, ¸(2007). "Robust scenario optimization based on downside-risk measure for multi-period portfolio selection". In: *OR Spectrum* 29, pp. 295–309.

Chaharsooghi, S., F. Momayezi, and N. Ghaffarinasab (2017). "An adaptive large neighborhood search heuristic for solving the reliable multiple allocation hub location problem under hub disruptions". In: *International Journal of Industrial Engineering Computations* 8, pp. 191–202.

Chiang, W. C. and R. A. Russell (1996). "Simulated annealing metaheuristics for the vehicle routing problem with time windows". In: *Annals of Operations Research* 63, pp. 3–27.

Christofides, N., A. Mingozzi, and P. Toth (1981). "State Space Relaxation for the Computation of Bounds to Routing Problems". In: *Networks* 11, pp. 145–164.

Clarke, G. and J. R. Wright (1964). "Scheduling of Vehicle Routing Problem from a Central Depot to a Number of Delivery Points". In: *Operations Research* 12, pp. 568–581.

Coelho, L. C., J.-F. Cordeau, and G. Laporte (2012). "Consistency in multi–vehicle inventory routing". In: *Transportation Research Part C: Emerging technologies* 24, pp. 270–287.

Cömert, S. E. et al. (2017). "A new approach for solution of vehicle routing problem with hard time window: an application in a supermarket chain". In: *Journal Sādhanā* 42, pp. 2067–2080.

Cowling, P. I., G. Kendall, and E. Soubeiga (2001). "A Hyperheuristic Approach to Scheduling a Sales Summit". In: *In: Burke, E., Erben, W. (eds.) PATAT 2000* 2079, 176—190.

Croes, G. A. (1958). "A method for solvmg the traveling salesman problems". In: *Operations Research* 6, pp. 791–812.

Dantzig, G. and J. Ramser (1959). "The Truck Dispatching Problem". In: *Management Science* 6, pp. 80–91.

Desrochers, M., J. Desroiers, and M. Solomon (1992). "A new optimization algorithm for the vehicle routing problem with time windows". In: *Operation Research* 342–354, pp. 80–91.

Dhahri, A. et al. (2016). "A VNS-based Heuristic for Solving the Vehicle Routing Problem with Time Windows and Vehicle Preventive Maintenance Constraints". In: *Procedia Computer Science* 80, pp. 1212–1222.

Diana, M. and M. Dessouky (2004). "A New Regret Insertion Heuristic for Solving Large-Scale Dial-a-Ride Problems with Time Windows". In: *Transportation Research Part B: Methodological* 38, pp. 539–557.

Drake, J., E. Özcan, and E. Burke (2012). "An Improved Choice Function Heuristic Selection for Cross Domain Heuristic Search". In: *Lecture Notes in Computer Science* 7492, pp. 307–316.

Dror, M., G. Laporte, and F. V. Louveaux (1993). "Vehicle Routing with Stochastic Demands and Restricted Failures". In: *ZOR – Methods and Models of Operations Research* 37, pp. 183–273.

Dror, M. and P. Trudeau (1986). "Stochastic vehicle routing with modified savings algorithm". In: *European Journal of Operational Research* 23, pp. 228–235.

Eksioglu, B., A. V. Vural, and A. Reisman (2009). "The vehicle routing problem: A taxonomic review". In: *Computers & Industrial Engineering* 57, pp. 1472–1483.

Errico, F. et al. (2016). "A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times". In: *European Journal of Operational Research* 249, pp. 55–66.

Fabozzi, F. J. et al. (2007). "Robust portfolio optimization and management". In: *Wiley*.

Fisher, M. L., K. O. Jornsten, and O. B. G. Madsen (1997). "Vehicle routing with time windows: two optimization algorithms". In: *Operations Research* 45.

Foisy, C. and J. Potvin (1993). "Implementing an insertion heuristic for vehicle routing on parallel hardware". In: *Computers & Operations Research* 20, pp. 737–745.

Gambardella, L. M., E. Taillard, and G. Agazzi (1999). "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows". In: *New Ideas in Optimization*, pp. 63–76.

Gehring, H. and J. Homberger (1999). "A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows". In: *Proceedings of EUROGEN99*, pp. 57–64.

Gendreau, M., G. Laporte, and R. Séguin (1996). "Stochastic vehicle routing". In: *European Journal of Operational Research* 88, pp. 3–12.

Goren, S. and I. Sabuncuoglu (2008). "Robustness and stability measures for scheduling: single-machine environment". In: *IIE Transactions* 40, pp. 66–83.

Gounaris, C.E., W. Wiesemann, and C.A. Floudas (2013). "The robust capacitated vehicle routing problem under demand uncertainty". In: *Operations Research* 61, pp. 677–693.

Gülpinar, N., D. Pachamanova, and E. ¸Canakoglu (2013). "Robust strategies for facility location under uncertainty". In: *European Journal of Operational Research* 255, pp. 21–35.

Gülpinar, N. and B. Rustem (2007). "Worst-case robust decisions for multi-period mean-variance portfolio optimization". In: *European Journal of Operational Research* 183, pp. 981–1000.

Gutin, G. and A. Punnen (2002). "The Traveling Salesman Problem and Its Variations". In:

Halse, K. (1992). "Modelling and Solving Complex Vehicle Routing Problems". In: *Ph.D., Institute of Mathematical Statistics and Operations Research, Technical University of Denmark* 60.

Hashimoto, H. et al. (2010). "Recent progress of local search in handling the time window constraints of the vehicle routing problem". In: *4OR's Journal* 8, pp. 352–60.

Hazir, Ö., M. Haouari, and E. Erel (2010). "Robust scheduling and robustness measures for the discrete time/cost trade-off problem". In: *European Journal of Operational Research* 207, pp. 633–643.

Hemmelmayer, V. C. (2014). "Sequential and Parallel Large Neighborhood Search Algorithms for the Periodic Location Routing Problem". In: *European Journal of Operational Research* 243, pp. 352–60.

I. Sungur, F. Ordóñez and M. Dessouky (2008). “A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty”. In: *IIE Transactions* 40, pp. 509–523.

Jornsten, K. O., O. B. G. Madsen, and B. Sorensen (1986). “Exact solution of the vehicle routing and scheduling problem with time windows by variable splitting”. In: *Technical Report, Department of Mathematical Modeling, Technical University of Denmark*.

Kallehauge, B. et al. (2005). “Vehicle Routing Problem with Time Windows”. In: *(eds) Column Generation* 1, pp. 67–98.

Kohl, N. (1995). “Exact methods for time constrained routing and scheduling problems”. In: *Phd. Thesis, Department of Mathematical Modeling, Technical University of Denmark*.

Kohl, N. and O. Madsen (1997). “An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean Relaxation”. In: *Operations Research* 45, pp. 395–406.

Kolen, A., A. Rinnooy, and H. Trienekens (1987). “Vehicle routing with time windows”. In: *Operations Research* 45, pp. 266–273.

Labadie, N., C. Prins, and C. Prodhon (2016). “Metaheuristics for Vehicle Routing Problems”. In: *WILEY* 3.

Lenstra, J. K. and A. H. G. R. Kan (1981). “Complexity of vehicle routing and scheduling problems”. In: *Networks* 11, pp. 221–227.

Lin, S. and B. W. Kernighan (1981). “An effective heuristic algorithm for the traveling-salesman problem”. In: *Operations Research* 21, pp. 498–516.

Madsen, O. B. G. (1988). “Variable splitting and vehicle routing problem with time windows”. In: *Technical Report 1A/1988, Department of Mathematical Modeling, Technical University of Denmark*.

Minoux, M. (2010). “Robust network optimization under polyhedral demand uncertainty isNP-hard”. In: *Discrete Applied Mathematics* 158, pp. 597–603.

Mladenović, N. and P. Hansen (1997). “Variable neighborhood search”. In: *Computers & Operations Research* 24, pp. 1097–1100.

Moghaddam, B. F., R. Ruiz, and S. J. Sadjadi (2012). “Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm”. In: *Computers & Industrial Engineering* 62, pp. 306–317.

Montoya-Torres, J. R. et al. (2015). "A literature review on the vehicle routing problem with multiple depots". In: *Computers & Industrial Engineering* 79, pp. 115–129.

Nasri, M., I. Hafidi, and A. Metrane (2020b). "Multithreading Parallel Robust Approach for the VRPTW with Uncertain Service and Travel Times". In: *Symmetry* 13.

Nasri, M. et al. (2020a). "A robust approach for solving a vehicle routing problem with time windows with uncertain service and travel times". In: *International Journal of Industrial Engineering Computations* 11, pp. 1–16.

Noorizadegan, M., L. Galli, and B. Chen (2012). "On the heterogeneous vehicle routing problem under demand uncertainty". In: *In: 21st International Symposium on Mathematical Programming, Berlin, Germany* 11, pp. 1–25.

Or, I. (1976). "Traveling salesman type combinatorial optimization problems and their relation to the logistics of blood banking". In: *Phd. Thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL.*

Pamela, J. Palomo-Martínez, M. Angélica Salazar-Aguilar, and G. Laporte (2017). "Planning a selective delivery schedule through Adaptive Large Neighborhood Search". In: *Computers & Industrial Engineering* 112, pp. 368–378.

Pillac, V. et al. (2013). "A parallel matheuristic for the technician routing and scheduling problem". In: *Optimization Letters* 7, pp. 1525–1535.

Pisinger, D. and S. Ropke (2007). "A general heuristic for vehicle routing problems". In: *Computers & Operations Research* 34, pp. 2403–2435.

Potvin, J. and J. M. Rousseau (1993). "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows". In: *European Journal of Operational Research* 66, p. 331.

Potvin, J. Y. et al. (1996). "The vehicle routing problem with time windows part I: Tabu search". In: *INFORMS Journal on Computing* 8, pp. 158–164.

Pradenas, L., B. Oportus, and V. Parada (2013). "Mitigation of greenhouse gas emissions in vehicle routing problems with backhauling". In: *Expert Systems with Applications* 40, 2985—2991.

PrescottGagnon, E., G. Desaulniers, and L. M. Rousseau (2009). "A branch and price based large neighborhood search algorithm for the vehicle routing problem with time windows". In: *Networks* 54, pp. 190–204.

Prins, C. (2004). "A simple and effective evolutionary algorithm for the vehicle routing problem". In: *Computers & Operations Research* 12, pp. 1984–2002.

Røpke, S. (2009). "PALNS-A software framework for parallel large neighborhood search". In: *In 8th Metaheuristic International Conference CDROM, Metaheuristic International Conference: Hamburg, Germany*.

Røpke, S. and D. Pisinger (2006). "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". In: *Transportation Science* 40, pp. 455–472.

Rouky, N. et al. (2018). "A Robust Metaheuristic for the Rail Shuttle Routing Problem with Uncertainty: A Real Case Study in the Le Havre Port". In: *The Asian Journal of Shipping and Logistics* 34, pp. 171–187.

Russell, R. (1995). "Hybrid heuristics for the vehicle routing problem with time windows". In: *Transportation Science* 29, pp. 156–166.

Salazar-Aguilar, M. Angélica, A. Langevin, and G. Laporte (2012). "Synchronized arc routing for snow plowing operations". In: *Computers & Operations Research* 39, pp. 1432–1440.

Santini, A., S. Ropke, and L.M. Hvattum (2018). "A Comparison of Acceptance Criteria for the Adaptive Large Neighbourhood Search Metaheuristic". In: *Journal of Heuristics* 24, pp. 783–815.

Savelsbergh, M. W. P. (1985). "Local search for routing problems with time windows". In: *Annals of Operations Research* 4, pp. 285–305.

Shaw, P. (1997). "A new local search algorithm providing high quality solutions to vehicle routing problems". In: *Technical Report*.

— (1998). "Using constraint programming and local search methods to solve vehicle routing problems". In: *In International conference on principles and practice of constraint programming. Springer Berlin Heidelberg* 1520, pp. 417–431.

Solomon, M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problem with time Window Constraints". In: *Operations Research Journal* 35, pp. 254–265.

Taillard, É. et al. (1997). "A tabu search heuristic for the vehicle routing problem with soft time windows". In: *Transportation Science* 31, pp. 170–186.

Talbi, E-G. (2013). "Combining metaheuristics with mathematical programming, constraint programming and machine learning". In: *Annals of Operations Research* 31, pp. 170–186.

Tan, X., X. Zhuo, and J. Zhang (2006). “Ant Colony System for Optimizing Vehicle Routing Problem with Time Windows (VRPTW)”. In: *In Computational Intelligence and Bioinformatics. ICIC 2006; Lecture Notes in Computer Science* 4115, pp. 33–38.

Tasan, A. S. and M. Gen (2012). “A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries”. In: *Computers & Industrial Engineering* 62, pp. 755–761.

Thangiah, S. (1995). “Vehicle routing with time windows using genetic algorithms”. In: *In Application Handbook of Genetic Algorithms: New Frontiers*, pp. 253–277.

Ticha, H. Ben et al. (2017). “Empirical analysis for the VRPTW with a multigraph representation for the road network”. In: *Computers & Operations Research* 88, pp. 103–116.

Toklu, N. E., R. Montemanni, and L. M. Gambardella (2013). “An ant colony system for the capacitated vehicle routing problem with uncertain travel costs”. In: *Swarm Intelligence (SIS), Proceeding IEEE Symposium on*, pp. 32–39.

Toth, P. and D. Vigo (2002). “An overview of vehicle routing problems”. In: *In 9 of SIAM Monographs on Discrete Mathematics and Applications; SIAM: Philadelphia*, pp. 1–26.

Vidal, T., G. Laporte, and P. Matl (2019). “A concise guide to existing and emerging vehicle routing problem variants”. In: *European Journal of Operational Research* 286, pp. 401–416.

Wu, L., M. Hifi, and H. Bederina (2017). “A new robust criterion for the vehicle routing problem with uncertain travel time”. In: *Computers & Industrial Engineering* 112, pp. 607–615.

Yousefi, H. et al. (2017). “Solving a bi-objective vehicle routing problem under uncertainty by a revised multi-choice goal programming approach”. In: *International Journal of Industrial Engineering Computations* 8, pp. 283–302.