Vehicle Routing Problems with road-network information

Hamza Ben Ticha

▶ **To cite this version:**

Hamza Ben Ticha. Vehicle Routing Problems with road-network information. Other [cs.OH]. Université Clermont Auvergne [2017-2020], 2017. English. NNT : 2017CLFAC071 . tel-01793702

HAL Id: tel-01793702
https://theses.hal.science/tel-01793702

Submitted on 16 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre NNT : 820

THÈSE DE DOCTORAT DE L'UNIVERSITÉ CLERMONT AUVERGNE

opérée au sein de

l'École des Mines de Saint-Étienne

École Doctorale

Sciences pour l'Ingénier de Clermont Ferrand

Spécialité de doctorat : Informatique

Soutenue publiquement le 20/11/2017, par :

Hamza BEN TICHA

Vehicle Routing Problems with Road-Network information

Devant le jury composé de :

Rapporteurs :	Frederic SEMET	Professeur, Ecole Centrale de Lille, France
	Claudia ARCHETTI	Professeur, Université de Brescia, Italie
Examinateurs :	Fabien LEHUEDE	Maître Assistant, Ecole des Mines de Nantes, France
	Thierry GARAIX	Maître Assistant, EMSE, France
	Tom VAN WOENSEL	Professeur, Université des Technologies d'Eindhoven, Pays-Bas
Directeur de thèse :	Nabil ABSI	Professeur, EMSE, France
Co-directeur de thèse :	Alain QUILLIOT	Professeur, ISIMA, France
Co-encadrant :	Dominique FEILLET	Professeur, EMSE, France

This work has been financed by the budget of LabEx IMobS3 backed by funding from a state aid and managed by the National Research Agency under the Investments for the Future program, an aid from the Union European via the European Regional Development Fund (FEDER - Auvergne Region) and an aid from the Auvergne Region.

La bourse est cofinancée par l'Union européenne. L'Europe s'engage en Auvergne avec le Fonds européen de développement régional

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	IMAGE, VISION, SIGNAL	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	X. Delorme, Maître assistant
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	CR	Génie industriel	CMP
AUGUSTO	Vincent	CR	Image, Vision, Signal	CIS
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BADEL	Pierre	MA(MDC)	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA(MDC)	Informatique	FAYOL
BLAYAC	Sylvain	MA(MDC)	Microélectronique	CMP
BOISSIER	Olivier	PR1	Informatique	FAYOL
BONNEFOY	Olivier	MA(MDC)	Génie des Procédés	SPIN
BORBELY	Andras	MR(DR2)	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BRUCHON	Julien	MA(MDC)	Mécanique et ingénierie	SMS
BURLAT	Patrick	PR1	Génie Industriel	FAYOL
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Image Vision Signal	CIS
DELAFOSSE	David	PR0	Sciences et génie des matériaux	SMS
DELORME	Xavier	MA(MDC)	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR1	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
DOUCE	Sandrine	PR2	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FAVERGEON	Loïc	CR	Génie des Procédés	SPIN
FEILLET	Dominique	PR1	Génie Industriel	CMP
FOREST	Valérie	MA(MDC)	Génie des Procédés	CIS
FOURNIER	Jacques	Ingénieur chercheur CEA	Microélectronique	CMP
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Génie des Procédés	SPIN
GAVET	Yann	MA(MDC)	Image Vision Signal	CIS
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GONDRAN	Natacha	MA(MDC)	Sciences et génie de l'environnement	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
NORTIER	Patrice	PR1		SPIN
OWENS	Rosin	MA(MDC)	Microélectronique	CMP
PERES	Véronique	MR	Génie des Procédés	SPIN
PICARD	Gauthier	MA(MDC)	Informatique	FAYOL
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR1	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Image Vision Signal	CIS
POURCHEZ	Jérémy	MR	Génie des Procédés	CIS
ROBISSON	Bruno	Ingénieur de recherche	Microélectronique	CMP
ROUSSY	Agnès	MA(MDC)	Génie industriel	CMP
ROUSTANT	Olivier	MA(MDC)	Mathématiques appliquées	FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	PR2	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR1	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

Mise à jour : 01/02/2016

Acknowledgements

This research project would not have been possible without the help and support of many people. In particular, I would like to thank my supervisors Nabil, Dominique and Alain. This document would not have been possible without their wise guidance and all their suggestions. They were always available for discussions and they supported me in all the different steps of the work. Working with them was always enjoyable and very pleasant. I will always be grateful to them.

I would like to thank Dr. Frederic Semet and Dr. Claudia Archetti for accepting the invitation of my committee and reviewing the present dissertation. I also want to thank Dr. Thierry Garaix and Dr. Fabien Lehuede for being part of my committee. In addition, I would like to thank Dr. Tom Van Woensel for his participation in the jury and for hosting me for three months in the Eindhoven university of technologies. I would like to thank him also for the interesting discussions that we had and for his precious contribution in some parts of this work.

I would like to thank all SFL members for the great moments and the interesting discussions that we had together. A special thank goes for Ali with whom I arrived in SFL at the same time. Although his *special* character, I was lucky to find a person like Ali on my way, with whom I could talk about everything, from technical, personal and professional problems. I would like to think Hamideh and Mehrdad for the gift that they made to me for my Thesis defense and for the funny discussions that we had. I want also to think Stephane, Agnes, Valeria, Claude and Pierre for their encouragement.

Finally, I would like to dedicate this thesis to my parents for whom I owe all what I achieved in my life, to my brothers and my little sister for their love and support throughout my studies and life. I want to think *ma cèhrie* Imen who believed on me and supported me during the moments of doubts.

Contents

Chapter 1

Introduction

Transportation is a central activity in logistics and supply chain management. It is estimated that distribution costs represent almost half of the logistic costs and can account up to 70% of the total cost of goods for some industries [33]. In 2009, the transport industry generated up to 7% of the Gross Domestic Product (GDP) of the European Union (EU) [101]. In the United States, the total logistics costs accounted up to 10% of the GDP for the year 2000 and the transportation on its own represented more than the half of this contribution on the GDP [69]. Besides, transportation has an huge environmental impact. According to Rodrigue [116], 60% of the global oil consumption and 25% of energy consumption are due to transport industry. The European Environment Agency indicates that, *in 2013, the transport sector contributed almost one-quarter (*24.4%*) of total EU greenhouse gas emissions* [46].

Therefore, improving the efficiency of transportation systems is a very important task to increase competitiveness and reduce the environmental impact of companies. This task gives a rise to the well-known Vehicle Routing Problem (VRP). This problem arises in many real-life applications that involve the optimization of passenger or freight transportation activities (*e.g.*, transportation of less-abled people, scheduling of school buses, mail delivery, waste collection, pickup and delivery of goods in the retail industry).

The VRP can be described as the process of designing a set of minimum-cost routes to be performed by a fleet of vehicles to serve a set of geographically dispersed customers. Since its introduction by Dantzig and Ramser [31] in 1959, the VRP has drawn many researchers' attention. Thousands of papers and books have been devoted to study optimization problems where routing issues are involved. Eksioglu et al. [51] enumerate approximately $1,500$ indexed publications on vehicle routing problems.

Conventionally, vehicle routing problems are defined on road networks in which service demand (pickup or delivery) points are associated with specific locations. Thus, the quality of the solution, when executed, depends largely on the quality of the representation of the road network. In the literature, most of approaches are, implicitly, based on a key assumption that, the best origin-destination path between two points of interest (*e.g.*, depot, customers) can be easily defined. Accordingly, vehicle routing problems are addressed using a complete graph, so-called customer-based graph, where a node is introduced for each point of

interest and an arc represents the best path in the road network between its endpoints. In many situations, the customer-based graph may fail to accurately represent the original road network and more information from the road network are needed to correctly address the routing problem.

A first situation is when several attributes are defined on road segments. In this case, defining the best path between two nodes in the road network involves a multi-objective optimization problem. Then, the solution of such a problem consists of the set of non-dominated paths. Considering only one path may result on an overestimation of the solution cost, since good solutions, perhaps optimal, could be discarded from the solution space [61, 89].

In some problems, road segment attributes depend on decisions to be fixed during the routing process. In such problems, it is difficult to define the best path between two points of interest and a precomputed customer-based graph can have a negative impact on the solution quality. It is the case, for example, of vehicle routing problems where fuel emissions has to be minimized. In this problem, the route cost depends on the travelling speeds along road segments. These travelling speeds are assumed to be defined by the traffic condition. It has been shown that by allowing the modification of travel speeds at any point of the road network, significant savings in fuel emissions can be obtained [109]. In such a setting, the best travel time and fuel consumption associated with a path between two points of interest cannot be computed in advance.

The customer-based graph represents also an oversimplified abstraction of the original road network, when the latter has a complex structure. In some urban areas, the access to some roads and zones is subject to time restrictions or tolls. Also, some travel regulations (U-turns, parking, etc.) should be considered when computing a VRP solution. All these details and information cannot be addressed when using the customer-based graph modeling.

Finally, from a methodological point of view transforming the vehicle routing problem on the road network into a standard vehicle routing problem could have a negative impact on computational performances. Although that the customer-based graph reduces the number of nodes compared to a graph representation of the original road network, it may substantially increase the set of arcs. Thus, it prevents from exploiting some properties, such as sparsity or planarity.

In the literature, the number of papers that investigate these issues is very limited. The few studies proposed in this regard do not examine the impact of the customer-based graph on the solution quality of the general vehicle routing problem and are, mostly, interested in solving specific problems.

One objective of this thesis is to point out the limits of the traditional complete graph representation and to confirm the need of new models and approaches with more information

from the road network in order to address correctly vehicle routing problems. For this aim, we focus on vehicle routing problems where several attributes are defined on road segments. As already mentioned, these problems are typically addressed via the customer-based graph where an arc is assumed to represent the best path between tow points of interest. However, the best path is unlikely to be the same for all attributes. Paths with different compromises are, thus, not considered in the customer-based graph. To handle this issue, two approaches are proposed in the literature. The first approach consists in representing the road network using a multigraph where an arc is introduced for every alternative path [61]. The second approach consists in solving the vehicle routing problem directly on a graph that mimics the original road network [89], that we call *road-network* graph. It is worth mentioning that in [61], authors are mainly interested in the development of an efficient solution method for a specific vehicle routing problem. While in [89], the objective is to show that it is more efficient to solve the vehicle routing problem directly on the road network graph than using a multigraph representation.

In this thesis, we start by exploring the multigraph representation of the road network. We investigate the construction of the multigraph and we propose an efficient solution method that computes the set of alternative paths for every pair of points of interest in the road network. Then, we investigate the tractability of such a modelling for vehicle routing problems and we analyse in depth the impact of this approach on the solution quality. In a second step, we focus on the development of an efficient solution method that can handle the multigraph setting. In a third step, we explore the road network based approach. We propose a complete comparison of the two approaches. We present an extensive computational study based on benchmark problems as well as instances derived from real data. In a final step, we are interested in problems where travel times vary over the time of the day, called time dependent vehicle routing problems. We explain that, in such settings, it is intractable to tackle the problem using the multigraph representation. We develop a branch-and-price algorithm to solve the problem directly on the road network graph. Finally, we analyse the impact of the proposed approach on the solution quality for time-dependent vehicle routing problems.

The following section gives the structure of the thesis and presents on overview of the contribution of each chapter.

Thesis structure

The thesis is made by eight chapters including this one. The next six chapters consist of six self-sufficient articles, of which one is published and three are submitted for publication. In the following, we present a brief description of the content of each chapter.

Chapter 2: Vehicle routing problems with road-network information: State of the art

Vehicle routing problems are, typically, addressed using a complete graph representation of the road network, so-called customer-based graph. In many situations, this representation can have significant (negative) effects on solution quality. In order to achieve a clear understanding on this subject, we survey the papers that evoke the limits of the customer-based graph and that propose alternative approaches to handle these limits. These papers are classified into four classes depending on the motivation that lead to proceeding differently than using the customer-based graph. For each class of papers, a discussion is proposed to identify the main directions for further research and development.

Chapter 3: A Solution Method for the Multi-destination Bi-objective Shortest Path Problem

In this chapter, we explore the construction of the multigraph representation for vehicle routing problems with several attributes on road segments. In this representation, an arc is introduced for each non-dominated path between two points of interest. The construction of the multigraph presents an important step in solving vehicle routing problems with the multigraph setting. This step involves a multi-objective Shortest Path Problem, which is NP-hard. It consists in computing the set of Pareto-optimal paths for each pair of nodes. This step could have significant impact on the efficiency of the multigraph-based approaches.

We propose a solution method that computes the set of Pareto-optimal paths from one to all other points of interest on a road network. Computational experiments based on instances derived from real road networks are carried out to evaluate the efficiency of the proposed algorithm. Obtained results show that multigraphs with 25 customers on a road network with more than 5400 nodes are computed in less than 1 second. The computing time reaches almost 4 minutes for instances with 500 customers on a road network with 19500 nodes.

Chapter 4: Empirical analysis for the VRPTW with a multigraph representation for the road network

This chapter investigates in depth the impact of the multigraph representation on the solution quality for vehicle routing problems on the road networks. It also examines the tractability of representing the road network with a multigraph. To do this, we consider the Vehicle Routing Problem with Time Windows (VRPTW) as a test-bed problem and we develop a branch-and-price algorithm that can handle the multigraph setting.

The main contribution of this chapter is that it analyses the impact of the multigraph representation based on an extensive experimental study. Results show that important savings in solution costs can be obtained for a large number of benchmark instances. We also observe that the multigraph setting slightly increase computing times compared to an equivalent branch-and-price scheme applied to two customer-based graphs where arcs represent, respectively the shortest paths and the fastest paths between each pair of points of interest.

Chapter 5: Adaptive Large Neighborhood Search for the Vehicle Routing Problem with Time Windows with a multigraph representation for the road network

This chapter is devoted to the development of an efficient solution method for the multigraph-based VRPTW. With the multigraph approach, some computational challenges arise. In particular, it has been shown that computing the optimal route for a given sequence of customers on the multigraph involves an NP-hard problem, so-called Fixed Sequence Arc Selection Problem (FSASP). Thus, even simple local search moves become difficult to evaluate.

We develop an adaptive large neighborhood search based algorithm that can efficiently address the multigraph setting. To handle the selection of arcs (FSASP), we propose a procedure based on an incremental data structure and a dynamic programming algorithm. This procedure permits to identify all non dominated subpaths that can connect a customer to the depot for a given sequence of customers. Then, using this information, insertion and removal of customers are evaluated efficiently. Experiments show that the proposed solution method provides near optimal solutions in reasonable computing times compared to the branch-and-price algorithm.

Chapter 6: A branch-and-price Algorithm for the Vehicle Routing Problem with Time Windows on a road-network graph

To handle the negative effects of the customer-based graph representation on the solution quality for vehicle routing problems with several attributes on road segments, two alternative approaches are investigated in the literature. The first approach consists in representing the road network with a multigraph. In the second approach, the problem is tackled directly on a graph that mimics the original road network.

The aim of this chapter is to evaluate the relative efficiency of these two modelling approaches. To do this, we develop a complete branch-and-price algorithm that can solve the problem on the road network graph. Note that using these settings, the standard branching rules cannot ensure the feasibility of the solution since a fractional solution can be sup-

ported by an integer flow matrix. To handle this issue, we propose a more suitable branching scheme. We compare solutions obtained using the proposed algorithm to those obtained using an equivalent multigraph-based branch-and-price algorithm. Computational experiments on instances derived from real-world road networks are performed in order to achieve comprehensive conclusions. The impact of different factors such as customers density in the road network, capacity constraints and time windows wideness, on the efficiency of the branch-and-price algorithms is analysed. Numerical results show that addressing the problem directly on the road network instead of using the multigraph representation decreases the quality of the lower bound and significantly increases the computing times.

This chapter has been done in cooperation with Tom van Woensel from the Eindhoven University of Technology, Eindhoven, Netherlands.

Chapter 7: The Time Dependent Vehicle Routing Problem with Time Windows and road network information

In this chapter, we are interested in the time Dependent vehicle routing problems where several attributes are defined on road segments. In these problems, travel times on road segments depend on the time of the day.

The aim of this chapter is to point out the limits of the traditional customer-based graph modeling for vehicle routing problems when travel times are time dependent. Since it is intractable to represent the road network using a multigraph in this setting (the set of alternative paths between two points of interest varies over the time), we propose to tackle the problem directly on the road network graph and develop a branch-and-price algorithm. In order to derive comprehensive conclusions, we also propose two algorithms that compute the two customer-based graphs: the time-dependent shortest path based graph and the time-dependent fastest path based graph. Obtained results illustrate the negative effects of the customer-based graph on the solution quality and the attractive savings that can be obtained. It comes out that by tackling the problem on the road network graph optimal solutions have been obtained for infeasible (due to time constraints) instances with the customer-based graph modeling.

This chapter has been done in cooperation with Tom van Woensel from the Eindhoven University of Technology, Eindhoven, Netherlands.

Chapter 8: Conclusions and perspectives

This chapter concludes the thesis and proposes directions for further research.

Chapter 2

Vehicle routing problems with road-network information: State of the art

This chapter is submitted for publication in *Networks* journal.

Abstract

Vehicle routing problems have drawn researchers' attention for more than fifty years. Most approaches found in the literature address these problems using the so-called customer-based graph, a complete graph representing the road network, where a node is introduced for every point of interest (e.g., customers, depot...) and an arc represents the best path between two points. In many situations, this representation induces negative effects on the solution quality or efficiency. A growing number of works in the literature investigate these issues and propose modeling taking account of more detailed information from the road-network. In this paper, we review these works and classify them with respect to the type of negative effects provoked by the customer-based graph.

Keywords: *Vehicle Routing Problem, Road Network, multigraph.*

2.1 Introduction

Vehicle routing problems (VRPs) constitute one of the most studied classes of combinatorial optimization problems in the operations research literature. Since their introduction by Dantzig and Ramser [31] in 1959, VRPs have drawn many researchers attention. Thousands of papers and books have been written about these problems [128, 65, 87, 132, 55]. One reason is the numerous applications where VRPs are involved in logistics, supply chain management, distribution systems, freight and passenger transportation systems, car navigation systems, etc. Many types of VRPs have been introduced to address different objectives or constraints, and a large number of solution approaches have been proposed in the literature (*e.g.*, [63], [107], [108], [59] and [117]).

A VRP can be described as the problem of computing an optimal set of routes to meet, entirely or partially, customer demands. Before pursuing, let us make clear that in this paper, we use the acronym VRP for any vehicle routing problem, not for the specific problem referred as Vehicle Routing Problem in the literature. For practical relevance, many constraints can be introduced such as customer time windows, limits on route lengths, the possibility to organize routes with multiple trips or to start from multiple depots, to name only a few.

An important issue in VRPs is the nature of customer demand. In most cases, each pickup or delivery point is considered separately and is identified with a specific location in the road network. In this case, the VRP is called a node routing problem. In another category of problems, that typically involve a large number of pickup or delivery points, the demand is aggregated at the level of road segments. These problems are then called arc routing problems. Applications can be found in mail delivery, garbage collection or snow removal, for example. A recent survey on these problems is presented by Corberàn and Laporte [23]. In this paper, we focus on the first category: the node routing problems.

In the literature, most approaches proposed for nodes routing problems are implicitly based on the key assumption that, for each pair of points of interest (*e.g.*, customers, depots), the best origin-destination path can be defined in advance. Then, the road network is transformed into a complete graph where a node is introduced for each point of interest and an arc represents the best path between its endpoints. The different attributes associated with an arc are then computed in accordance with this path. Following Huang *et al.* [71], we call this graph *customer-based graph* in the rest of the paper. In many situations, modeling the problem with a customer-based graph can have important consequences:

- A first situation is when several attributes are defined on road segments (travel cost, distance, travel time, etc.). In this case, the problem of finding the best path between two points of interest is multicriteria by nature. Then, efficient paths, with different compromises between the different attributes, are missed when constructing the customer-based graph. Good solutions, perhaps optimal, can be discarded from the solution space [61, 85, 8].

- Another difficulty might come from the computation of shortest paths for a given attribute. In some cases, like when fuel consumption minimization is searched for, the value of an attribute (the cost here) can be a complex function of several parameters as starting time or load. This difficulty is even reinforced when decisions can be taken at the scale of road segments. It is the case, for example, when instructions can be given to drivers to optimize their speed (*e.g.*, Qian and Eglese [109]). Computing the customer-based graph can then simply become untractable.

- The customer-based graph is also sometimes an oversimplified abstraction of the road network, when the latter has a complex structure, as in urban areas. In some cities, for example, the access to some areas is subject to time restrictions or tolls. Also, the

detail of travel restrictions (parking, U-turns, etc.) cannot be apprehended with the customer-based graph.

- From a methodological point of view, finally, it is not so obvious that adopting the customer-based graph always has a positive impact on computational performances. Compared to a graph representation of the road network, using the customer-based graph enables a potentially very large drop of the node set size, but at the expense of a possibly large increase of the arc set. Furthermore, it prevents from exploiting some proprieties, such as sparsity or planarity.

Recently, an increasing number of studies have raised the aforementioned issues and have started investigating VRPs by considering more information from the road network. In this paper, we overview these works and we outline the proposed alternative approaches. We organize this survey with respect to the four situations depicted above. First, we focus, in Section 2.2, on the literature related to VRPs with several attributes. In Section 2.3, we analyze works that deal with *complex* attributes. In Section 2.4, we review studies that consider complex road-network structures. Finally in Section 2.5, we survey papers interested in the computational performance of the graph representation.

2.2 Vehicle routing problems with several attributes on road segments

In practical applications of VRPs, arcs in the road network are often labeled with several attributes, as: distance, time, monetary cost, energy consumption (for electric vehicles), scenic interest (for sightseeing), hazard, travel time robustness... These attributes can be needed to ensure route feasibility, regarding operational constraints, and/or to evaluate the solution quality. As already mentioned, these VRPs are typically addressed using a customer-based graph where an arc is assumed to represent the best path in the road network between its two endpoints. However, the best path for an attribute is not necessarily the same for the others, *e.g.*, the fastest path in an urban network is unlikely to be the same as the shortest path. Considering only one path between two nodes discards trade-offs between the attributes and may consequently discard good solutions from the solution space. Thus, addressing the problem with a customer-based graph can have negative impacts on the solution value.

In the literature, two alternative approaches have been investigated to address this issue. The first approach consists in representing the road network with a multigraph. In the second approach, the problem is directly solved on a graph that mimics the original road network and that we call *road-network* graph. In the next two subsections, we review these two approaches.

2.2.1 Multigraph

A first possible way to keep trace of all efficient paths between every pair of nodes is to represent the road network with a multigraph. In this representation, an arc is introduced for every efficient (with respect to the considered attributes) path in the original network.

As far as we know, Garaix *et al.* [61] were the first to point out that using a customer-based graph when several attributes are associated with road segments could lead to an over-estimation of the solution cost. They investigate a Dial-a-Ride Problem, motivated by a real-world On-Demand Transportation System for a rural area. They propose to address the problem using a multigraph in order to efficiently handle the cost-time trade-off. They pursue two main objectives: evaluating the tractability of this approach and demonstrating its benefits against using a traditional customer-based graph. Several authors have built on these results and addressed the same issues since, for different problems: Lai *et al.* [85] for a time-constrained mixed-fleet VRP, Ben Ticha *et al.* [8, 7] for the Vehicle Routing Problem with Time-Windows (VRPTW), Huang *et al.* [71] for a time-dependent time-constrained VRP. We review below the contributions of these papers, with the section organized according to the two aforementioned issues: tractability and benefits.

Tractability of the multigraph

Garaix *et al.* [61] underline an important consequence of the multigraph. They notice that a route can no longer be described by a sequence of visited customers. Even with this sequence known, the problem of selecting an arc between each pair of consecutive customers is NP-hard (it can be expressed as a multidimensional multiple choice knapsack problem). They call this problem the Fixed Sequence Arc Selection Problem (FSASP).

They propose a pseudo-polynomial dynamic programming algorithm for the solution of the FSASP. Though the method is relatively fast, it is not compatible with local search methods where a lot of neighbor solutions should be explored. Starting from this statement, they develop a simple greedy best insertion method followed by a descent, in which few FSASPs have to be solved.

Lai *et al.* [85] propose a tabu search heuristic. In order to deal with the additional complexity induced by the selection of arcs, the authors propose to evaluate each move heuristically. Given a move, and the associated customer sequence, they tackle the FSASP by a fast and effective greedy method inspired by existing methods for the Knapsack Problem.

Ben Ticha *et al.* [7] develop a heuristic method based on an Adaptive Large Neighborhood Search. To handle the selection of arcs (FSASP) when exploring neighbor solutions, they improve the dynamic programming algorithm presented by Garaix *et al.* [61]. Given a current solution, and before starting exploring neighbor solutions, they apply a forward and

a backward dynamic programming algorithms to identify, for each customer, all the non-dominated labels that can connect the customer to the depot. Using this information, the evaluation of the insertion or the removal of customers is significantly accelerated.

Besides their heuristic approach, Garaix *et al.* [61] propose a branch-and-price algorithm. The interest is that using a multigraph does not add any algorithmic complexity: as long as the pricing problem is a constrained shortest path problem addressed with a labeling algorithm, one just has to consider all the outgoing arcs when extending labels. They however underline the negative impact on computing times. More recently, Ben Ticha *et al.* [8] also implemented a branch-and-price to evaluate the benefits of using a multigraph. Strangely, contrary to Garaix *et al.* [61], they only observe a slight increase in computing times compared to an equivalent branch-and-price applied on two customer-based graphs: the *min-cost* graph where the best path is selected according to the distance, and the *min-time* graph, where it is selected according to travel times.

Benefits obtained with the multigraph

Garaix *et al.* [61], Lai *et al.* [85], Ben Ticha *et al.* [8] and Huang *et al.* [71] are all particularly interested in evaluating the benefits that can be achieved with the multigraph. They all conduct extensive computational analyses to assess these benefits. An intersting point is that all these studies are performed on different problems and with very different types of instances. They all arrive to the same conclusion: important benefits can be obtained.

This is not the purpose of this survey to detail all these experiments and analyze the relevance of the different managerial insights that they provide, but one can just mention that they typically report average benefits between 5 to 15% against optimal solutions computed on a customer-based graph. One should also emphasize that most experiments are carried out with instances obtained from real urban, suburban or rural road networks, with up to thousands of nodes.

Construction and characteristics of the multigraph

A difficulty implied by the use of a multigraph is the construction of the multigraph itself. Building a multigraph requires solving multicriteria shortest path problems, which is NP-hard. This step might then *a priori* be detrimental to the use of this kind of approaches.

In Garaix *et al.* [61], this step is only slightly commented. The authors indicate that their multigraphs are obtained by dynamic programming, with an algorithm similar to the one they use in their branch-and-price scheme (for the pricing problem). Lai *et al.* [85] consider a multigraph with only two arcs in parallel. Huang *et al.* [71] also only introduce a subset of non-dominated paths. These paths are the min-distance path, plus a series of min-time paths computed with different starting times (in their problem, travel times are time-dependent).

Ben Ticha *et al.* [8] discuss this issue in more details. They quickly explain how the multigraph can be constructed efficiently with an adaptation of Dijkstra and A^* algorithms, and refer to a technical paper (Ben Ticha *et al.* [9]) for more details. Multigraphs are obtained in a few seconds for instances with 75 customers and a road network with about 5000 nodes. Around 5 minutes are needed when the size of the road network grows up to 20000 nodes.

Ben Ticha *et al.* [8] also report information on the number of parallel arcs in the multigraphs they compute. For different graph topologies and different types of correlation between two attributes, they never report more than 3 or 4 parallel arcs on average. It might sound small at first sight, but, after all, considering everybody's own real-life experience, one can hardly imagine more than a few reasonable possible paths when traveling between two given locations.

2.2.2 Road-network graph

The second approach to conserve the entire solution space is to tackle directly VRPs on a road-network graph, that is, a graph that mimics the road network. Basically, arcs correspond to road segments and nodes to the extremities of these segments.

A key paper in this regard is proposed by Letchford *et al.* [89]. They revisit the branch-and-price approach presented by Garaix *et al.* [61]. They decide to use as a test-bed problem the multiple Traveling Salesman Problem with Time Windows (m-TSPTW). They are guided by the idea that it might be more efficient to tackle the problem on the road-network graph than to introduce the multigraph. Especially, they point out that, in the worst case, constructing and storing the multigraph could be exponential in time and in space (which was however later empirically denied in [8, 9] – as seen in Section 2.2.1). Hence, in their opinion, it sounds more natural to keep the initial road network structure.

The authors identify two consequences on the branch-and-price scheme. First, in the pricing problem, shortest paths are searched in the road-network graph and it is possible to traverse a customer node without executing the service. If a dynamic programming algorithm is used, two labels are then needed when reaching a customer node: one label that serves the customer and one label that just passes through the location of the customer. Second, in the branching scheme, standard rules cannot be applied for the same reasons as those encountered when solving arc routing problems by branch-and-price: the flow matrix derived from a solution can be integer while the solution is fractional. The authors implement the pricing problem but leave the branching scheme for future researches. Computational results demonstrate the interest of their approach when computing the linear relaxation of the problem, compared with an equivalent column generation procedure applied to the multigraph.

Ben Ticha *et al.* [11] reproduce this approach for the VRPTW and additionally implement the branching scheme. When the situation of a factional solution with an integer flow support happens, they define two branches: in the first branch they enumerate all the feasible routes restricted to the support graph and apply a branch-and-bound; in the second branch they force using at least one arc not in the flow support graph. However, they empirically notice than this situation hardly ever happens. They evaluate in depth the relative efficiency of applying the branch-and-price on the multigraph or on the road-network graph, and, contrary to Letchford *et al.* [89], conclude on the superiority of the former, for which they observe significant smaller computing times. Furthermore, they explain that addressing the problem with a road-network graph decreases the quality of the lower bound if the route structure allows multiple services to customers. Indeed, in the road-network graph it is possible to come back to a customer at a very low price (while in the customer-based graph, one needs to visit at least one intermediate customer).

In connection with the study presented in [89], a stream of papers concerned with VRPs on road networks focus on the so-called Steiner Traveling Salesman Problem (STSP). The aim of the STSP is to find a min-cost cycle visiting a set of required nodes in a road-network graph. This problem was introduced, independently, by Orloff [103], Fleischmann [56] and Cornuéjols *et al.* [29]. In [56] and [29], the STSP is formulated as an integer program with a linear number of variables and an exponential number of constraints, and is then solved using a cutting plane method. In addition, Fleischmann [56] proposes a way to extend the solution procedure to the VRP, but reveals the difficulties that would arise in this case. Recently, Letchford *et al.* [90] proposed compact formulations for the STSP, with a linear number of variables and constraints.

A few authors define their VRP on a road-network graph, but switch to a customer-based graph within their (heuristic) solution procedure. Huang *et al.* [70] are interested in the planning of tourist sight-seeing itineraries. A tourist has to visit a given subset of locations in a road network. Each road segment is defined by four attributes related to four objectives: travel time, vehicle operating cost, safety level and surrounding scenic view quality. The authors simply aggregate the four attributes, which permits to compute best paths between points of interest and to construct a customer-based graph. Zografos and Androutsopoulos [137] develop a decision support system for hazardous material transportation and emergency service unit location. They face a VRP that seeks for the best set of vehicle routes for the distribution of hazardous materials in terms of cost and risk minimization. To each road segment is assigned a travel time and a risk measure. Again, the authors aggregate these two attributes and switch to the customer-based graph.

It is finally worth mentioning that a large part of the literature using road-network graphs concerns Arc Routing Problems. We do not review this literature here and we redirect the reader to [23] and [44] for recent books. We report however that solution methods devel-

oped for the Capacitated Arc Routing Problem (CARP) and its variants should be an important source of inspiration when addressing node-routing VRPs on road-network graphs (as demonstrated by Letchford and Oukil [91] and Bode and Irnich [12] for branch-and-price algorithms).

2.2.3 Related works using a multigraph

Besides the papers surveyed in Section 2.2.1, that introduce multigraphs to deal with the multiple attributes found on road segments, a few other papers introduce multigraphs with other motivations. Because VRPs modeled with multigraphs are rare, we include these papers in this survey.

As far as we know, Baldacci *et al.* [3] were the first to introduce a multigraph in the context of a vehicle routing problem. They are interested in a Multiple Disposal Facilities and Multiple Inventory Locations Rollon-Rolloff Vehicle Routing Problem where the objective is to define a transportation plan for trailers between customers, disposal facilities and inventory locations. The authors show that the considered problem is equivalent to a time constrained VRP on a multigraph. In this representation, nodes correspond to feasible trips (a trip is a sequence of movements of the tractor to perform a service); parallel arcs between two nodes represent the set of possible vehicle moves and are defined with different cost-time compromises.

Caramia and Guerriero [17] study a long-haul freight transportation problem. Their motivation stems from a real-life application implying multimodal routes. Their objective is to define a transportation plan that minimizes the route cost and travel time, and that maximizes a transportation-mean sharing index. The authors observe that two nodes could be connected with several alternative routes: some arcs represent the same transportation mode but have different traveling times and costs, and some arcs represent different transportation modes. Consequently, the transportation network could be represented using a multimodal multigraph. They show that this structure enlarges significantly the solution space and makes the definition of the best solution for the multi-objective scenario very difficult. A heuristic method based on a local search scheme is developed to solve the problem.

In the two above papers, multiple arcs offer multiple possibilities with different compromises when moving between nodes. Though the contexts are very different, the authors are eventually confronted to the same issues as those mentioned in Section 2.2.1.

This not true for Wang and Lee [133] that introduce the so-called Time-Dependent Alternative Vehicle Routing Problem. The TDAVRP is a vehicle routing problem with time windows, time-dependent travel times and in which nodes are connected with two arcs. The first arc represents the best path when the traffic is low. It is defined with a time-dependent

speed distribution. The other arc represents an alternative path for peak hours. This arc is not sensitive to traffic and is assigned a constant travel time. A heuristic algorithm based on a Particle Swarm Optimization is developed. In the case of this problem, the selection of an arc does not induce different compromises. Instead, it just relies on the actual departure time: depending on this time an arc dominates the other.

The same remark holds for Setak *et al.* [122]. They introduce the so-called Time Dependent Pollution Routing Problem in Multigraph (TDVRPM). In the TDVRMP, several arcs exist between each pair of customers, each arc representing a possible path in the road network. The objective is to minimize a complex combination of carbon emission, driver wages and tolls. However, the best path between two customers can be found in advance when the starting time and the vehicle load are known. They address the problem with a tabu search heuristic.

Finally, Reinhardt *et al.* [113] also investigate a VRP with a multigraph support. The authors introduce a new generalization of the VRPTW in which additional fixed costs are associated with subsets of edges. These costs for example correspond to fees that give access to some parts of the road network. The problem is modeled with a multigraph. More details are given on this work in Section 2.4.

2.2.4 Discussion

The aforementioned papers state the limits of the customer-based graph representation for VRPs when several attributes are defined on road segments. Computational analyses all show that the potential impact on solution costs is far from being negligible. Conducting experiments with industrial VRP software and on real data would be very interesting to confirm, or not, these conclusions. It is indeed to be noted that the objective function in software typically merge distance (vehicle cost) and time (driver cost), and that these attributes may also be combined when the best paths needed to construct the customer-based graph are computed.

To replace the customer-based graph, a multigraph is used in some works and in others the problem is tackled on a graph similar to the original road network. In this context, many research directions can be explored. First, it would be interesting to investigate more in depth the solution of VRPs on road-network graphs. Letchford *et al.* [89] suggest that it could be better to use these types of graphs than using multigraphs, but they only partially elaborate on this issue and are essentially contradicted by Ben Ticha *et al.* [11]. Further works are needed both for the development of exact and heuristic approaches on road-network graphs. Certainly, the literature on arc routing problems should be an important source of inspiration here. We however would like to insist on the fundamental difference between the latter and node-routing VRPs on road-network graphs: a demand expressed on a typically large subset

of arcs (generally almost all of them) versus a demand represented with a typically small subset of nodes (a few among thousands).

Also, several questions remain open for multigraphs. Is this modeling tractable in more complicated VRP settings? Is it always possible to manage efficiently the construction of the multigraph, the selection of arcs in solution methods? What are the worst-cases in the number of parallel arcs?

2.3 Vehicle Routing Problems with *"complex"* attributes

The most part of the research on VRPs assumes that all attributes in the road network can easily be computed in advance. These attributes are, typically, given as input or computed using given information, *e.g.*, travel times may be computed using average speeds on road segments. Based on this assumption, when the VRP involves a single attribute on road network arcs, the best path between two points of interest can easily be defined and the VRP can be tackled with a customer-based graph.

However, there are some situations where computing the best path for a given attribute between two points of interest is not that simple. An easy example is that of time-dependent travel times. In theory, it is easy to precompute the min-time path for every possible starting time. It then enables constructing a customer-based graph with travel time functions associated with every arc. One just have to be conscious that the road-network path associated with an arc in this graph might change by time of the day. Even if in practice it can be more complicated (because of memory or accuracy issues related to the modeling of travel time data, as discussed in Eglese *et al.* [47]), this case does not fail into the topic of this survey.

Further complications arise with time-dependent cost functions. These problems are typically found for fuel consumption or carbon emission minimization, with time-dependent travel times. The shortest-cost path problem between two points of interest becomes non trivial if Bellman's principle of optimality does not hold, *i.e.*, if it might be preferable to arrive at some intermediate node with a higher cost (but earlier) to avoid congestion and save some future cost (see, *e.g.*, Wen *et al.* [134]). An additional difficulty is even faced if fuel consumption is modeled as depending on the vehicle load. In these situations, the construction of a customer-based graph becomes less and less viable.

Even more complicated, there are some situations where the decision-maker has the possibility to take decisions that could influence the itinerary to follow between two points of interests. We have found two examples in the literature. In the first example, it is possible to voluntarily decrease speed to limit fuel consumption. In the second example, one has to manage driver breaks (to respect working hour regulations).

In this section, we will successively survey the three cases evoked above: the case of time-dependent cost functions, the case when speed is a decision variable and the case when working hour regulations are considered.

Note that the interest in such complex problems is a consequence of the recent technical developments that allow collecting accurate information on travel times and speeds, by time of day or day of week. Having this information enhances the need for a better management of temporal issues in routing optimization software.

2.3.1 Complex time-dependent cost functions

Wen and Eglese [135] address a time-dependent VRP with time windows and a cost function composed of fuel cost, driver cost and congestion charge. The fuel and driver costs are time-dependent. Congestion charge consists in a fee that is paid when entering a given zone. It is due at most once for a vehicle. As explained in Section 2.2, a trade-off exists between time and cost, but the authors ignore this trade-off. They first take aside congestion charges and construct a customer-based graph with (time-dependent) min-cost paths between points of interest. For that matter, they use a heuristic method developed in Wen *et al.* [134]. They then take congestion charges into account as follows. A second (time-dependent) cost matrix is computed for which entering the zone that requires a congestion charge is not allowed. Virtually, it means that an alternative path is defined for each pair of points of interest, and the resulting two matrices can be interpreted as a multigraph with two alternative arcs. However, it is not considered as such by the authors in the tabu search approach that they propose. Indeed, they notice that, given a vehicle route in a solution: either the congestion charge will be paid and the first matrix should be used to evaluate the route cost or it would not be paid and the second matrix should be used. Hence, these two options are considered and the best one is kept.

Ehmke *et al.* [49] study a time-dependent VRP with the objective of minimizing carbon emissions. In their model, emissions not only depend on the vehicle speed but also on the vehicle load. For that reason, they consider that the complete set of min-cost paths cannot be precomputed. They explicitly define their VRP on the road-network graph. They propose to address it with a tabu search approach. Local search moves are standard except that min-cost paths between points of interest, for given starting time and load, are computed when evaluating a move, if necessary. A heuristic algorithm proposed in Ehmke *et al.* [48] is used for that purpose. The best paths are stored to avoid repeating the same computations. Furthermore, the authors show and exploit the following property: when the min-cost path is the same for an empty and a full-loaded vehicle, it will remain the same for any intermediate load. They use this property in preprocess by precomputing all those paths that will not depend on the vehicle load.

2.3.2 Speed optimization

Qian and Eglese [109] study the problem of finding the route with minimal carbon emissions that visits a predefined sequence of customers subject to time windows. The objective is to specify the path followed by the vehicle between each pair of successive customers, together with the travel speeds and waiting times along these paths. The authors address the problem on a time-dependent road-network graph and allow adapting speed/stopping at any point of the network. In particular, speed is allowed to be lower than the speed limit defined by traffic conditions. With this setting, the optimal path between two successive customers, and the associated attributes, cannot be precomputed (at least easily). Actually, to solve the problem, the authors propose two solution approaches: a dynamic programming algorithm and a heuristic adaptive search method that first selects a set of candidate paths and then defines the travel speed for each road segment on the selected paths.

In a subsequent study [110], the authors are interested in the Time Dependent Vehicle Routing Problem with Time Windows (TDVRPTW). The problem investigated in [109] is then the subproblem met when evaluating carbon emissions for a given vehicle route. For this reason, they still have to address the problem on the road-network graph. They propose a tabu-search-based column generation algorithm. At each iteration, a tabu search heuristic is used to generate new columns (routes). Travel speeds and carbon emissions for each new route are optimized using the heuristic method proposed in [109].

2.3.3 Driver working hour regulation

Chassaing *et al.* [19] propose to consider working time and break times of drivers in the TDVRP. They propose to address the problem directly on the road network where break-times are allowed at any node. Two objectives are considered: minimizing the working time and the riding time. The authors observe that due to the traffic conditions, it could be interesting to wait at some node in the road network (break-time) in order to avoid congestion at peak hours. This decision may lead to a shorter ridding time but involves a longer working time. To handle this trade-off, they propose to consider waiting-times as decision variables. They develop a genetic algorithm based on a giant tour decomposition where the path to be used between two consecutive customers is determined when extending the label from the first endpoint and the possibility of inserting break-times is considered when evaluating a path.

2.3.4 Discussion

The number of papers that investigate vehicle routing problems with "*complex*" attributes has grown recently. Different issues (time-dependent data, travel speed regulation, complex cost functions, working and break times regulation, etc.) question using a customer-based graph. These works should certainly be pursued.

A complementary recent trend is to introduce stochastic information. Travel times or costs are indeed stochastic by nature and a lot of recent papers explore this issue. As far as we know, the preexistence of a customer-based graph with the associated stochastic information is almost always assumed in these papers (*e.g.*, [88], [93], [81]). One noticeable exception is by Huang *et al.* [71] (see Section 2.2.1) that consider stochastic travel times and investigate the following two-stage stochastic strategy. At first stage, traditional vehicle routes (sequences of customers) are defined, based on average travel times. Then, at the second stage, when information on travel times is known, the path actually followed in the road network is selected. In Ehmke *et al.* [49] also (see Section 2.3.1) speed information is taken from a sample and can be considered as stochastic. There is clearly still a lot to investigate about the relationship between stochastic models and road-networks graphs.

Another related promising research direction is that of vehicle routing problems with real-time traffic information. Due to recent advances in information and communication technologies, actual traffic conditions are known in real time and "exact" travel times can be obtained at any time of the day. Again, the principle of using a customer-based graph could be questioned in this context. We are not aware of any paper on this subject.

It is finally also interesting to highlight the case of urban distribution with electric vehicles. Energy consumption by an electric vehicle heavily relies on many parameters as distance, speed, slope and vehicle load. While the first papers in this area voluntarily simplified the energy consumption models, the trend is now to move to more realistic models (Pelletier *et al.* [106]) and better consider the so-called range anxiety. One can anticipate future investigations similar to those that already started about carbon emissions.

2.4 Vehicle routing problems on complex road-networks

Goods transportation constitutes an important activity in urban areas. Many recent studies in the VRP literature focus on optimizing transport and distribution activities in urban centers, the so-called city logistics. In most of these studies, the problem is defined on a customer-based graph. Three types of decisions are mainly addressed: assignment and sequencing decisions which define the customer visit order for each vehicle and scheduling decisions which determine the visit timetable. Implicitly, they assume that the path selected between

two customers does not depend on the sequence. In other words, they assume a complete independence in the path selection decisions. However, this assumption is not always relevant in complex road networks as those found in urban areas. Indeed, the way a customer is reached might influence the way it will be left. In the context of vehicle routes with many stops and small distances between stops, the impact can be important. A second type of dependencies exists when portions of the network are subject to fees. Depending on whether the fees are paid or not, the paths will not be the same.

The literature is very limited on this subject. We only found three papers that investigate alternatives to the customer-based graph: one in regard to the modeling of vehicle stops, two in regard to fees. Two of these papers are reviewed below, the third one, Wen and Eglese [135], having already been surveyed in Section 2.3.1.

2.4.1 Fine modeling of vehicle stops

Lang *et al.* [86] investigate the impact of considering alternative parking points at customer locations. They study the VRPTW with the objective of minimizing fuel consumption. They address the problem on a realistic urban road network where bi-directional roadways are considered. They point out that due to the medial strips between lanes of traffic with opposite directions, vehicles may have to travel an additional distance to serve customers across the road. To handle this issue, they introduce alternative parking points, at which the vehicle can stop and from which the deliveryman can cross the street and walk to the customer. These alternative stopping points are introduced according to the load to be delivered and the distance to walk. Customer time windows and service times are updated accordingly.

The authors address the problem with an ant colony algorithm. They however do not detail how the alternative stopping points are handled in the algorithm. Apparently, these points are simply managed as other points and included in the candidate set (with their own probabilities) when deciding of the next destination for an ant. A case study is carried out on a road network in the south of Beijing. A test problem with 10 customers was generated. Results show the impact of considering alternative stop points on the quality of the solution: the total traveled distance is reduced by 2.5% and the total fuel consumption is reduced by 4.5%.

2.4.2 Access with fees

Reinhardt *et al.* [113] investigate VRPs in which fees must be paid for accessing roads, areas or bridges. These fees can reflect payment for toll roads, ferry connections, investment in new facilities or the need for certifications to access to war zones or areas of unrest. Then, connection costs between customer locations do not depend only on traveling costs, but also

rely on other decisions, basically buying or not access to a set of routes in the transportation network. Hence, a standard customer-based graph does not model correctly the problem. Reinhardt *et al.* [113] consider the VRPTW and propose an extension that take fees into account, which they call Edge Set VRPTW. In this new problem, edges are regrouped in different subsets, each subset being characterized by a fixed cost. Once this cost is paid, all the vehicles can access all the edges of the associated subset. The authors explain that the problem should be modeled with a multigraph, in order to differentiate between edges that connect the same pair of nodes but do not belong to the same edge subsets. However, in this study, they only address the special case when the graph is simple, *i.e.*, only one edge exists between each pair of nodes. They justify this assumption by contexts where the fastest path should always be preferred (as, hazardous materials transportation). They propose a branch-cut-and-price algorithm. The adaptation of this algorithm to the general (multigraph) case is only slightly discussed and left as a perspective.

2.4.3 Discussion

Research on vehicle routing problems on road networks with complex structures is very limited. We only found a few papers about stopping points and fees. Furthermore, these papers propose very preliminary results. Extensive computational study and further analyses are needed to draw comprehensive conclusions.

As far as we know, also, many issues still remain to be investigated, such as restrictions on street access, hours of operations, etc. It is worth mentioning that these issues have been considered in the context of shortest path problems (*e.g.*, U-turn restrictions [136, 1, 92], route access and lane changing restrictions [82]) and in arc routing problems [23]. When dealing with the optimization of urban vehicle routes with numerous stops, they certainly deserve more attention.

2.5 Impact on solution method efficiency

2.5.1 Size of models

In real life, vehicles travel on a road network where they have to serve a limited subset of locations. Since the first paper by Dantzig and Ramser [31], most approaches for vehicle routing optimziation have modeled this situation with the customer-based graph. Transforming the road network into a customer-based graph is commonly admitted as being more efficient as it precomputes once for all the best path between every pair of points of interest. This intuition can however be questioned.

The first to explore this issue was Fleischmann [56]. He claims that nodes in road networks typically have small degrees and that the number of arcs in a road-network graph can be largely inferior to that in the corresponding customer-based graph. When addressing the problem with a flow model, it can critically increase the number of variables. The author first considers the optimization of the vehicle tour in a road-network graph for a single vehicle, *i.e.*, the STSP (see Section 2.2.2). He proposes a solution method based on a cutting planes scheme. In the same paper, the author also suggests to extend the solution procedure to the VRP and points out the main difficulties that would arise in this case.

Recently, Letchford *et al.* [90] investigated more in depth the impact of considering the road network when modeling the TSP. They propose different mathematical formulations for the STSP with polynomial numbers of variables and constraints. Computational results show that instances with up to 200 required nodes are solved using a standard mathematical programming solver.

2.5.2 Relationship with arc routing problems

Another source of reflection about the relative efficiency of the different graphs can be found in the arc routing literature. Several authors investigate the possibility of transforming arc routing problems into node routing problems. Other authors prefer addressing these problems directly on a road-network. Detailing all these attempts is out of the scope of this paper, let us just develop one significant example. Letchford *et al.* [91] study the Capacitated Arc Routing Problem (CARP). They recall that the most efficient approach so far for solving exactly the CARP consists in transforming the problem into a node routing problem, via series of shortest path computations, and then in applying any efficient exact solution method for the resulting VRP. They detail the main drawbacks of this approach: it increases the size of the graph, it removes special structures that might appear in the road network, it increases symmetries. Following this discussion, they develop a branch-cut-and-price algorithm directly on the road-network and explain the interest of this approach.

One should however repeat here the main difference between arc routing problems and VRPs on road-network. In arc routing problems, a large proportion of arcs usually require service. In VRPS, only a very small proportion of nodes (in the road-network graph) has to be visited. Hence, the impact of transforming the road network into a customer-based graph is hardly comparable.

2.5.3 Discussion

Papers cited above investigate the impact of the traditional customer-based graph representation on solution method performance. They suggest that transforming the original road network into a customer-based graph is likely to increase the number of variables in a mathe-

matical formulation and hence, affect negatively the efficiency of solution algorithms. However, they are far from giving definitive proofs on this issue.

The literature on arc routing problems would tend to support this claim, but the nature of the problems are different. Probably the best graph representation is largely influenced by the characteristic of the instance: special structure of the road-network, density of customers...

A topic that is largely ignored in the VRP literature is possibility to take advantage of a road-network graph in heuristic methods. Of course, this is implicitly done when, for example, extending a vehicle route to its nearest neighbor or applying a best insertion. But one could imagine a more explicit exploitation of the road-network graph, *e.g.*, by applying local search moves to the routes represented in this graph.

2.6 Conclusion

Since its introduction by Dantzig et Ramser [31], the Vehicle Routing Problem has been the subject of intensive research efforts. Thousands of papers and books have been written about VRPs and in which numerous applications have been addressed. Most of the approaches proposed in the literature are based on a key assumption that the best path between each pair of points of interest can be easily defined, and hence, the problem can be tackled using a customer-based graph representation. In many situations, this representation could have negative effect on the solution quality or efficiency.

The literature review presented in this paper confirms that there are many reasons to consider the solution of vehicle routing problems with more detailed information from the road network. Some papers investigate the limits of the traditional customer-based graph representation on the solution quality when several attributes are defined on road segments or when arc attributes are somehow "complex" to compute in advance. To handle these limits, two modeling approaches are examined, namely the multigraph representation and working directly on a road-network graph. Other papers focus on the impact of the complex structure of road networks on the solution of vehicle routing problems. Finally, some other papers emphasize the interest of solving vehicle routing problems directly on the road network from a methodological point a view.

In spite of the increasing interest in proceeding differently than using the customer-based graph, the number of papers that investigate these issues still remains relatively limited. Many questions and research topics are still unexplored. Several of these topics have been discussed in length all along this survey, with respect to many contexts such as urban logistics, electric vehicle routing, stochastic VRPs or dynamic VRPs, among others. We hope that the theoretical and practical perspectives it opens will be an important source of inspiration for future works on VRPs.

Chapter 3

A solution method for the Multi-destination Bi-objectives Shortest Path Problem

This chapter is submitted for publication in *INFORMS Journal on Computing*.

Abstract

This paper is devoted to study the multi-destination bi-objective shortest path problem. This problem consists in finding the set of Pareto-optimal paths linking a set of points of interest in a network. Our motivation stems from the data preprocessing for vehicle routing problems on road networks. We propose a solution method based on a labeling approach with a multi-objective A search strategy that can handle the multi-destination case. Computational results based on instances generated from real road networks show the efficiency of the proposed algorithm compared to state-of-art approaches.*

Keywords: *Multi-destination, Bi-objective shortest path problem, Pareto-optimal paths, labeling algorithm, A* algorithm.*

3.1 Introduction

The Vehicle Routing Problem (VRP) can be described as the problem of designing a set of routes that starts and ends at a depot and that visits a number of geographically dispersed locations, called customers. In the standard version of the problem, a node is introduced for each point of interest (the depot or a customer) and a distance matrix is used to report information on the shortest path between every pair of nodes. Methods with a polynomial complexity to compute such a matrix are well-known. These methods aim at computing the shortest origin-destination path, such as the goal directed search called also A* (see [68]) or at computing one-to-all shortest paths (from a source node to all other nodes), such as Dijkstra's algorithm (see [42]) or at computing all-to-all shortest paths, such as Floyd's algorithm (see [58]).

In many real-world applications, several attributes on road segments have to be considered when tackling the VRP and customer nodes may be connected using many different

paths with different trade-offs. In such situations, the distance matrix could be insufficient to correctly model the problem [61, 89]. An alternative modeling approach proposed by Garaix et al. [61] consists in representing the road network using a multigraph. In this representation, an arc is introduced for every non-dominated path linking two customer locations. In this case, the data preprocessing step for the VRP involves a Multi-objective Shortest Path Problem and consists in computing the set of Pareto-optimal paths for each pair of nodes.

In this paper, we investigate the construction of the multigraph representation for vehicle routing problems where two attributes are associated with road segments. We propose a solution method that computes the set of bi-objective shortest paths from one to all other points of interest in a road network. The proposed approach is based on a dynamic programming approach with an A* algorithm. The A* is a single objective shortest path algorithm based on a guided search strategy. Basically, it expands first partial paths leading the most quickly to the destination node [68]. In this study, we propose to implement a bi-objective A* algorithm that handles the multiple destination case, *i.e.*, the search strategy selects, at each iteration, the partial path that could lead quickly to a non-dominated path given it will have to reach one of the destination nodes. To show the efficiency of the proposed approach, we conduct computational experiments based on several test problems and we compare obtained results to those obtained using classical labeling methods.

The rest of this paper is organized as follows: In Section 3.2, we present an overview of the related literature. Section 3.3 presents the basic concepts of the studied problem. In Section 3.4, we describe the proposed solution method. In section 3.5, we propose some enhancements for the developed algorithm based on time windows defined for the points of interest (source and destination nodes). Finally, numerical results are reported in Section 3.6.

3.2 Literature review

The Bi-objective Shortest Path Problem (BSPP) is an extension of the classical Shortest Path Problem (SPP) and belongs to the class of multi-objective combinatorial optimization problems. Given a network where two attributes are associated with arcs, the BSPP consists in determining the set of non-dominated paths that optimizes the two objective functions. The BSPP is an NP-hard problem [121]. Hansen [67] showed that the generation of Pareto-optimal solutions for the BSPP is intractable in the worst case, *i.e.*, the number of Pareto-optimal paths can be exponential in the number of nodes in the network.

The BSP problems have been widely studied. The most recent survey on Multi-objective Shortest Path Problems (MSPPs) is by Clímaco and Pascoal [22]. They were interested in exact solution approaches and their applications especially for routing problems in telecommunication networks. The annotated bibliography on multi-objective combinatorial opti-

mization presented by Ehrgott and Gandibleux [50] includes a section about shortest path problems. An overview of solution methods used to solve BSP problems is proposed by Skriver [125].

Basically, two main classes of exact solution methods are used to solve BSP problems: labelling methods that consists in enumerating the non-dominated labels associated with every node in the network and ranking methods which are based on single objective k-shortest paths methods.

Labeling methods for the MSPP are similar to their single objective counterparts. They are also partitioned into label setting and label correcting methods. In label setting algorithms, one label arriving at a known node is set as permanent at each iteration, (*e.g.*, [96], [67]). In label correcting algorithms, all labels are temporarily maintained during the search procedure and only non-dominated labels at the last iteration become permanent (*e.g.*, [28], [16]). Label correcting algorithms differ in the used selection strategy. Brumbaugh-Smith and Shier [16] used node-selection strategies where at each iteration a node i is selected and all labels arriving at i are extended through all outgoing arcs. Tung and Chew [131] introduced a different selection strategy in which all labels are treated separately. They proposed to select at each iteration a label at some node i and extend it through all outgoing arcs.

Martins and Santos [98] investigated labeling-based algorithms for the MSPP where arcs are given with arbitrary costs. They showed that for any network with no absorbent cycle the MSPP is bounded and proved the correctness of labeling methods. Guerriero and Musmanno [66] studied label setting and label correcting algorithms and introduced new label-selection and node-selection strategies. Computational experiments carried out with random and grid networks showed that node-selection methods are generally faster but for some instances label-selection methods were more suitable. More recently, Paixão and Santos [104] proposed a computational study of labeling methods used to solve the MSPP. They investigated 27 variants of the labeling algorithm where label correcting and label setting were combined. A large set of test problems consisting of more than 9000 instances was used to evaluate the different implementations. They stated that the obtained results show that the label correcting algorithm produces the fastest performance.

Many techniques for labeling algorithms were proposed to speed-up multi-objective shortest paths computation. Steward and White [127] proposed an extension of the goal directed algorithm, called also A^* algorithm, for the multi-objective scenario. Raith [111] used a stopping condition so-called "dominance by early results". It is based on the ascertainment that any label dominated by one of the labels at the destination node can not lead to a Pareto-optimal path. Demeyer et al. [36] proposed a similar stopping criterion and introduced a bi-directional search procedure.

Ranking methods consist in listing paths with a non-decreasing order of one of the ob-

jectives [21]. From this list, a set of non-dominated paths is selected. Martins [95] proposed a ranking method based on a deletion algorithm. In this method, the shortest path according to a selected objective function is determined at each iteration. This path is, then, eliminated from the network at the start of the next iteration. The algorithm stops when the k-shortest path is determined or when no path is found in the network.

Huarng et al. [72] showed that for the BSPP the ranking method proposed by Climaco and Martins [21] is not competitive with labeling approaches. Their computational results suggest that both label setting and label correcting algorithms are far better than k-shortest path approaches.

Another approach was proposed by Mote et al. [99] to solve BSP problems. This approach consists in a two-phase method. In the first phase, extreme solutions in the convex hull of the solution space are computed by solving the LP relaxation of the problem. In the second phase, an enumerative method is used to determine the set of Pareto-optimal paths. The main idea of this approach is that it takes advantage of the problem structure and extracts information on extreme points of the convex hull of the solution space in the first phase. This information is, then, used in the second phase to restrict, and thus, to speed up the enumerative method. Raith and Ehrgott [112] studied the two-phase approach. They investigated different combinations of methods used in the two phases: a network simplex method, single objective label setting and label correcting algorithms were tested in phase 1 and, ranking and bi-objective labeling approaches were explored in phase 2. They compared the two-phase method with purely labeling approaches and a ranking method. Computational experiments carried out on three different set of instances showed the competitiveness of the two-phase method with the different configurations. In their conclusions, Raith and Ehrgott [112] noticed that the efficiency of the solution methods depends a lot on the network structure and the choice of the approach, that could perform well, relies on the network type.

Most of the aforementioned studies assume that all criteria are additive. Other types of criteria have been explored such as the bottleneck type, *i.e.*, min-max or max-min functions. Gandibleux et al. [60] revisited Martins' algorithm for the MSPP [96] and proposed a generalization to the case where a combination of additive objectives and one bottleneck objective are considered. De Lima Pinto et al. [34] proposed an algorithm that can handle tricriterion shortest path problems with at least two bottleneck objective functions. Path deviation procedures proposed by Martins et al. [97] were also used to rank paths for the MSPP.

Besides these exact methods, many of well known heuristics have been applied to MSP problems such as evolutionary algorithms (*e.g.*, [105]), multi-objective ant colony optimization based algorithms (*e.g.*, [64]), etc. Tsaggouris and Zaroliagis [130] presented an improved Fully Polynomial Time Approximation Scheme for the MSPP that can find a set of approximatively Pareto-optimal paths in a polynomial time.

From a practical point of view, Müller-Hannemann and Weihe [100] studied the cardinality of the set of Pareto-optimal solutions for MSP problems. They showed that due to some characteristics occurring in various applications, the number of efficient solutions is small enough and can be bounded with a small constant. An illustration was presented for railway networks with two and three objectives.

This paper makes a number of contributions to the literature. First, we propose a solution method that can efficiently handle the multi-destination setting. As far as we know, there is no previous paper that investigates such a setting. Almost all proposed approaches aim at computing either one-to-one paths or one-to-all paths. Although the multi-destination case can be addressed using an algorithm for one of the two other cases (by applying one-to-one algorithm for every destination node or by considering only paths arriving at required nodes in a one-to-all solution), interesting properties arise in this case and could be exploited to solve the problem with lower computational efforts (see Section 3.3). The developed algorithm is based on a dynamic programming approach with a modified A* algorithm. The A* is a single objective shortest path algorithm. We propose to implement a multi-objective A* algorithm that can handle the multi-destination setting. Second, the multi-destination multi-objective shortest path problem presents potential interest in many real-life applications, especially in vehicle routing. We propose a tool that can compute optimal solutions for large sized problem in reasonable computing times.

3.3 Basic concepts

We define a network as a directed graph $G = (V, A)$ where V is a set of n nodes and A is a set of m arcs. Each arc (i, j) is given two non-negative values (d_{ij}, t_{ij}) representing the costs for the two objectives. Costs d_{ij} and t_{ij} correspond respectively to the distance and the travelling time in our application. Let $v_0 \in V$ denotes the source node and $C = \{v_1, v_2, ..., v_{n_c}\} \subset V$ the subset of required nodes, *i.e.*, the set of n_c destination nodes.

We define a path p in G from a node $u_0 \in V$ to a node $u_i \in V$ as an ordered list of nodes $P = (u_0, u_1, ..., u_i)$ such that $(u_k, u_{k+1}) \in A \ \forall k \in \{0, ..., i-1\}$. The cost vector of path P is the sum of arc costs; $(d(P), t(P)) = (\sum_{k=0}^{k=i-1} d_{u_k u_{k+1}}, \sum_{k=0}^{k=i-1} t_{u_k u_{k+1}})$.

The algorithm presented in this paper is based on a labeling approach. A label represents a subpath from a source node v_0 to a certain node $u \in V$. We define a label with the following information $L = (u, d(L), t(L))$ with u is the ending node of the subpath represented by L and, $d(L)$ and $t(L)$ correspond, respectively, to the total distance and the total travel time associated with the subpath represented by L.

In the following, we recall some concepts that will be used to compare labels [98, 112,

66].

Definition 3.1. *Dominance*
A vector (a_1, a_2) dominates a vector (b_1, b_2) if and only if $a_1 \leq b_1$ and $a_2 \leq b_2$ with at least one inequality being strict.

Consequently, a label L_1 dominates a label L_2 if and only if the cost vector $(d(L_1), t(L_1))$ dominates $(d(L_2), t(L_2))$. In the same way, a path p_1 dominates a path p_2 if and only if the vector $(d(p_1), t(p_1))$ dominates the vector $(d(p_2), t(p_2))$.

Definition 3.2. *Lexicographic order*
A vector $a = (a_1, a_2)$ is lexicographically smaller than a vector $b = (b_1, b_2)$, denoted by $a \underset{lex}{<} b$, if either $a_1 < b_1$ or both $a_1 = b_1$ and $a_2 < b_2$.

The relationship established in Definition 2 is reflexive, transitive and anti-symmetric. We use this relationship to define a total lexicographic ordering on paths and on labels. A path p_1 is lexicographically smaller than a path p_2, denoted by $p_1 \underset{lex}{<} p_2$, if and only if $(d(p_1), t(p_1)) \underset{lex}{<} (d(p_2), t(p_2))$. Similarly, a label L_1, is lexicographically smaller than L_2, denoted by $L_1 \underset{lex}{<} L_2$, if and only if $(d(L_1), t(L_1)) \underset{lex}{<} (d(L_2), t(L_2))$.

Definition 3.3. *Pareto-optimality*
A path p is Pareto-optimal if and only if there exists no feasible path p' that dominates p

Let $\mathcal{P}(u, v)$ denotes the set of all paths linking node u to node v in G and let $\mathcal{P}_{opt}(u, v) = \{p_1, p_2, ..., p_r\} \subset P(u, v)$ the set of Pareto-optimal paths. Paths in $\mathcal{P}_{opt}(u, v)$ are sorted according to the lexicographic order, *i.e.*, $p_1 \underset{lex}{<} p_2 \underset{lex}{<} ... \underset{lex}{<} p_r$. Moreover, paths in $\mathcal{P}_{opt}(u, v)$ are such that

$$d(p_1) < d(p_2) < ... < d(p_{r-1}) < d(p_r) \tag{3.1}$$

$$t(p_1) > t(p_2) > ... > t(p_{r-1}) > t(p_r) \tag{3.2}$$

Using this notation, we can easily prove the following properties:

Property 3.1. *$p_1 \in \mathcal{P}_{opt}(u, v)$ is the shortest path in distance from u to v in G, i.e., $d(p_1) = \min_{p \in P(u,v)} d(p)$.*

Property 3.2. *$p_r \in \mathcal{P}_{opt}(u, v)$ is the shortest path in time from u to v in G, i.e., $t(p_r) = \min_{p \in P(u,v)} t(p)$.*

Indeed, for each path $p \in \mathcal{P}(u, v) \setminus \mathcal{P}_{opt}(u, v)$ there exists at least one path $p_k \in \mathcal{P}_{opt}(u, v)$ such that p_k dominates p, *i.e.*, $d(p_k) \leq d(p)$ and $t(p_k) \leq t(p)$. Consequently, for each path $p \in \mathcal{P}(u, v)$, we have $d(p_1) \leq d(p)$ and $t(p_r) \leq t(p)$.

In the remainder of this paper, we denote by $(d^{inf}(u,v), t^{sup}(u,v))$ the cost vector associated with the shortest path in distance in $\mathcal{P}_{opt}(u,v)$ and we denote by $(d^{sup}(u,v), t^{inf}(u,v))$ the cost vector associated with the shortest path in time in $\mathcal{P}_{opt}(u,v)$.

We introduce a new measure, denoted $D(p)$, that indicates the detour in distance of the path $p \in P(u,v)$ compared to the shortest path in distance: $D(p)$ is given by:

$$D(p) = d(p) - d^{inf}(u,v) \geq 0 \tag{3.3}$$

Using this notation, we can define bounds on detour values associated with paths in $\mathcal{P}_{opt}(u,v)$:

Property 3.3. *For every path $p \in \mathcal{P}_{opt}(u,v)$, $0 \leq D(p) \leq D^{sup}$ holds, with $D^{sup} = d^{sup}(u,v) - d^{inf}(u,v)$*

Therefore, paths in $\mathcal{P}_{opt}(u,v)$ are also sorted in a non-decreasing order of $D(p)$, *i.e.*, $0 = D(p_1) < D(p_2) < ... < D(p_{r-1}) < D(p_r) = D^{sup}$.

3.4 Multi-destination Bi-objective Shortest Path Problem

In this section, we explore how to compute the bi-objective shortest paths from the source node v_0 to the destination nodes in $C = \{v_1, v_2, ..., v_{n_c}\} \subset V$ in the network $G = (V, A)$. Note that our approach can be easily extended to both one-to-one and one-to-all cases: by setting n_c to 1 for the first case and by setting n_c to $n-1$ for the second case. It is however specifically designed for the case with $1 < n_c < n - 1$.

In Section 3.4.1, we present the main ideas of our solution method for the single destination ($C = \{v_1\}$) case, then, we present a generalization of our algorithm for the multiple destination case. In Section 3.4.2, we prove that the proposed algorithm correctly provides the expected Pareto-optimal paths and we finally present some complexity analysis.

3.4.1 Solution method

The solution method proposed in this paper is similar to the label setting algorithm proposed, first, by Martins [96] which is, in turn, based on Dijkstra's algorithm [42]. The label setting algorithm for the MSPP is a straightforward extension of the single objective version. The main difference is that several labels arriving at a node u may be maintained during the search procedure, each representing a subpath from the source node to node u. At each iteration, the lexicographic smallest label $L = (u, d(L), t(L))$ among labels with respect to all nodes is selected. This label is, then, extended using all outgoing arcs (u,v). The extension of $L = (u, d(L), t(L))$ using arc (u,v) results in label $L' = (v, d(L) + d_{uv}, t(L) + t_{uv})$. Dominated

labels are eliminated from the obtained labels and from labels already present at node v. The algorithm terminates once all labels are processed.

Recall that labeling algorithms for the MSPP are supported by an adaptation of the Principle of Optimality for the SPP [96], which states that every non-dominated path is formed by non-dominated subpaths. In a label setting algorithm, these subpaths are represented by non-dominated labels and are considered in a lexicographic order. Hence, Pareto-optimal paths arriving at the destination node are provided during the labeling procedure. However, the algorithm can not terminate until all labels are processed. This issue can be handled using a stopping criteria that interrupts the search once all non-dominated paths at the destination node are generated.

In our approach, we propose to guide the search using an A^* strategy. Basically, the A^* algorithm constructs the shortest path by expanding, first, the subpaths that appear to lead more quickly to the best solution [68]. These subpaths are selected based on an estimate of the cost still to go to the destination node. In our implementation, we propose to select, at each iteration, among all temporary labels the one that may lead to a non-dominated path with the shortest distance. In other words, we select at each iteration the label $L = (u, d(L), t(L))$ that minimizes $d(L) + d^{inf}(u, v_1)$. In this way, paths arriving at the destination node are generated in a non-decreasing order of distance. The search can be terminated once the selected label is such that $d(L) + d^{inf}(u, v_1) > d^{sup}(v_0, v_1)$. Due to properties 3.1 and 3.2 (see Section 3.3), it is guaranteed that all non-dominated paths at the destination node v_1 have been found once the stopping criterion is met. Note that the computation of values $d^{inf}(u, v_1)$ and $d^{sup}(v_0, v_1)$ can be preprocessed, as detailed subsequently.

To handle the case of multiple destination nodes, we propose to adapt the search scheme such that the selection procedure takes into account all possible final destinations for each label. The selection function, then, aims at defining the label apt to lead the most quickly to a non-dominated path at a certain destination node. To do this, we select the label $L = (u, d(L), t(L))$ that minimizes the value:

$$\min_{1 \leq i \leq n} (d(L) + d^{inf}(u, v_i) - d^{inf}(v_0, v_i)) \tag{3.4}$$

We denote by $K(L)$ this value and call it the *key* of the label. It indicates the minimum detour in distance of the associated subpath given it will have to reach one of the destinations in C.

Using this search procedure, paths reaching each destination node are generated in a non-decreasing order of detour in distance $D(p)$. The algorithm should terminate once labels $L_i = (v_i, d^{sup}(v_0, v_i), t^{inf}(v_0, v_i))$ have been generated for every destination node $v_i \in \{v_1, ..., v_n\}$. Therefore, the stopping criterion is that the key of the selected label is larger than $\max_{1 \leq i \leq n} K_i^{sup}$ with $K_i^{sup} = d^{sup}(v_0, v_i) - d^{inf}(v_0, v_i)$.

Note that, to compute the key of a label $L = (u, d(L), t(L))$, distances $d^{inf}(u, v_i)$ and $d^{inf}(v_0, v_i)$ associated, respectively, with shortest paths in distance from v to v_i and from v_0 to v_i are needed. Also, distances $d^{sup}(v_0, v_i)$ (used to compute K_i^{sup}) associated with the shortest paths in time from v_0 to v_i are needed. These distances are determined in preprocessing as follows:

- Using a Dijkstra algorithm, we compute shortest paths in distance and in time from node v_0 to all nodes $u \in V$. Four tables are constructed: $d^{inf}(v_0, u)$ and $t^{sup}(v_0, u)$ indicating distances and times associated with shortest paths in distance and, $d^{sup}(v_0, v)$ and $t^{inf}(v_0, v)$ indicating distances and times associated with shortest paths in time.
- Using backward Dijkstra algorithms from all nodes $v_1, ..., v_n$, we compute shortest paths in distance and in time from all nodes $u \in V$ to destination nodes $v_i \in \{v_1, ..., v_n\}$. Four series of tables are obtained: $d^{inf}(v, v_i)$ and $t^{sup}(v, v_i)$ indicating distances and times associated with shortest paths in distance and, $d^{sup}(v, v_i)$ and $t^{inf}(v, v_i)$ indicating distances and times associated with shortest paths in time.

Note that in both forward and backward Dijkstra's algorithms, labels are considered in a lexicographical order established using the operator $\underset{lex}{<}$ (see Definition 3.2). Accordingly, we have the guarantee that all the computed paths are efficient.

The general scheme of the proposed solution method is described in Algorithm 3.1. We call this algorithm **multi-A* algorithm**. Theorem 3.1 proves that all Pareto-optimal paths from the source node v_0 to every destination node $v_i \in \{v_1, ..., v_{n_c}\}$ are generated when the algorithm terminates.

3.4.2 Optimality proof and complexity analysis

Theorem 3.1. *Optimality*
*All Pareto-optimal paths from source node v_0 to all destination nodes v_1,...,v_n are found once the **multi-A* algorithm** terminates.*

Proof. Let us assume that at the end of the **Algorithm 3.1** there exists a non-dominated path $p_{v_0v_k}$ from the source node v_0 to a destination node v_k that has not been found by the algorithm.

Let $(u_0, u_1, ..., u_r)$ be the sequence of nodes visited along the path $p_{v_0v_k}$ with $u_0 = v_0$ and $u_r = v_k$. Let us denote by $L_i = (u_i, d(L_i), t(L_i))$ the label associated with the subpath defined by the sequences of nodes $(u_0, ..., u_i)$ for all $i \in \{0, ..., r\}$. Due to the Principle of Optimality for the MSPP [96], when extending a label L_i along the arc (u_i, u_{i+1}) the obtained label L_{i+1} can not be dominated by any label in $Labels[u_{i+1}]$. Consequently, $p_{v_0v_k}$ is not found by **Algorithm 3.1** if and only if there exists $j \in \{1, ..., r-1\}$ such that $K(L_j) > \underset{1 \le i \le n}{max} K_i^{sup}$.

Algorithm 3.1 multi-A* algorithm for the BSPP

Preprocessing
Compute $d^{inf}(v_0, v)$, $t^{sup}(v_0, v)$, $d^{sup}(v_0, v)$ and $t^{inf}(v_0, v)$ for all $v \in V$
for all $v_i \in \{v_1, ..., v_n\}$ **do**
    Compute $d^{inf}(v, v_i)$, $t^{sup}(v, v_i)$, $d^{sup}(v, v_i)$ and $t^{inf}(v, v_i)$ for all $v \in V$
    Compute $K_i^{sup} = d^{sup}(v_0, v_i) - d^{inf}(v_0, v_i)$
end for
End Preprocessing
$L = (v_0, 0, 0)$
$Labels[v_0].add(L)$
$allLabels.add(L)$
while $K(allLabels.Min()) \leq \max_{1 \leq i \leq n} K_i^{sup}$ **do**
    $L = allLabels.extractMin()$;
    for all $(u, v) \in A$ **do**
        $L' = (v, d(L) + d_{uv}, t(L) + t_{uv})$
        if L' is not dominated by a label in $Labels[v]$ **then**
            $allLabels.add(L')$
            $Labels[v].add(L')$
        end if
        if a label $L'' \in Labels[v]$ is dominated by L' **then**
            $allLabels.remove(L'')$
            $Labels[v].remove(L'')$
        end if
    end for
end while
return $Labels[v_i]$ for all $v_i \in \{v_1, ..., v_n\}$

From properties 3.1 and 3.2, we know that for every non-dominated path p_{uv_k} from u to v_k the associated distance $d(p_{u_j v_k})$ verifies

$$d^{inf}(u, v_k) \leq d(p_{uv_k}) \leq d^{sup}(u, v_k). \tag{3.5}$$

Let $d(u_j, v_k)$ denotes the distance associated with the subpath visiting the sequence $(u_j, u_1, ..., u_r)$ with $u_r = v_k$. The distance $d(p_{uv_k})$ associated with the path p_{uv_k} is then given by $d(p_{uv_k}) = d(L_j) + d(u_j, v_k)$ for all $j \in \{1, ..., r-1\}$.

As $d^{inf}(u_j, v_k) \leq d(u_j, v_k)$ for all $j \in \{1, ..., r-1\}$, we have:

$$d(L_j) + d^{inf}(u_j, v_k) \leq d(p_{v_0 v_k}) \leq d^{sup}(v_0, v_k) \tag{3.6}$$

This implies that for all $j \in \{1, ..., r-1\}$

$$d(L_j) + d^{inf}(u_j, v_k) - d^{inf}(v_0, v_k) \leq d(p_{v_0 v_k}) - d^{inf}(v_0, v_k) \leq d^{sup}(v_0, v_k) - d^{inf}(v_0, v_k) \tag{3.7}$$

Since $K_k^{sup} = d^{sup}(v_0, v_k) - d^{inf}(v_0, v_k) \leq \underset{1 \leq i \leq n}{max} K_i^{sup}$ and $K(L_j) = \underset{1 \leq i \leq n}{min}\{d(L_j) + d^{inf}(u_j, v_i) - d^{inf}(v_0, v_i)\}$, for all $j \in \{1, ..., r-1\}$ $K(L_j)$ verifies:

$$K(L_j) \leq \underset{1 \leq i \leq n}{max} K_i^{sup} \tag{3.8}$$

□

The implementation of Algorithm 3.1 relies on two main data structures:

1. A priority queue *allLabels* is used to store the temporary labels and is based on $K(L)$ values. The following methods are used to manage labels in *allLabels*:
   - *Min*() returns the label L at the top of the priority queue, *i.e.*, with the smallest $K(L)$ value;
   - *extractMin*() extracts the label L at the top of the priority queue. Its complexity is in $O(log(|allLabels|))$. $|allLabels|$ is the number of maintained labels in *allLabels*;
   - *add*(L) inserts the label L into *allLabels*. Its complexity is in $O(log(|allLabels|))$;
   - *remove*(L) removes the label L from *allLabels*. Its complexity is in $O(|allLabels|)$.
2. A list $Labels[u]$ is attached to each node $u \in V$ in which labels arriving at u are stored. These lists are controlled using the following methods:
   - *add*(L) adds the label L to $Labels[u]$. This is done in $O(1)$;
   - *remove*(L) removes the label L from the list in $O(|Labels[u]|)$ with $|Labels[u]|$ is the size of $Labels[u]$

Hansen [67] showed that the label setting algorithm is pseudo-polynomial, *i.e.*, the number of operations performed by the algorithm is bounded by a polynomial in the characteristics of the problem and the magnitude of data. In the following we show that the **multi-A* algorithm** is also pseudo-polynomial and we give an approximation of its complexity.

Without loss of generality, we assume that data are integer (if it is not the case data can be multiplied by a power of ten without any impact on the theoretical complexity). Let $\delta(v)$ denotes $\underset{1\leq i\leq n}{max}(d^{sup}(v_0,v_i)-d^{inf}(v_0,v))-d^{inf}(v,v_i))$ for every node $v \in V$ and Δ denotes $\underset{v\in V}{max}(\delta(v))$.

Theorem 3.2. *Complexity*
The complexity of the ***multi-A* algorithm*** *is in* $O(m\Delta^2 log(n\Delta))$.

Proof. A label $L = (v, d(L), t(L))$ arriving at node v could lead to a Pareto-optimal path at a destination node $v_i \in C$ if $d(L) + d^{inf}(v, v_i) \leq d^{sup}(v_0, v_i)$. Therefore every label $L = (v, d(L), t(L))$ maintained at node v during the labeling procedure verifies $d^{inf}(v_0, v) \leq d(L) \leq d^{sup}(v_0, v_i)-d^{inf}(v, v_i)$. Thus, the number of these labels is bounded by $\delta(v) = \underset{1\leq i\leq n}{max}(d^{sup}(v_0, v_i)-d^{inf}(v_0, v)) - d^{inf}(v, v_i))$ (since data are assumed to be integer) and the total number of labels in the heap *allLabels* is bounded by $\sum_{v\in V} \delta(v) \leq n\Delta$.

The complexities of the main procedures performed during the search scheme are as follows:

- The extraction of the label with smallest key value requires $O(log(n\Delta))$ operations;
- For each new label $L' = (u, d(L'), t(L'))$, the dominance check which implies the insertion of the new (non-dominated) label in the list $Labels[u]$ and the removal of dominated labels from $Labels[u]$ and from the heap *allLabels*, requires $O(\delta(u)log(n\Delta))$ operations.
- The extension of a selected label $L = (v, d(L), t(L))$ through all outgoing arcs $(v, u) \in A$ requires $O(\sum_{(v,u)\in A}(1 + log(n\Delta) + \delta(u)log(n\Delta))$ operations: the first term in the sum corresponds to the generation of the new label, the second term corresponds to the insertion of the new label into the heap and the third term corresponds to the dominance check.

Since every label arriving at a node v and that is in the heap is selected once and $O(\sum_{(v,u)\in A}(1 + log(n\Delta) + \delta(u)log(n\Delta))) \leq O(\sum_{(v,u)\in A} \delta(u)log(n\Delta)$, the total complexity of the algorithm is given by:

$$O\left(\sum_{v\in V}\sum_{L\in Labels[v]}\left(log(n\Delta) + \sum_{u\in V;(v,u)\in A}\delta(u)log(n\Delta)\right)\right)$$

$$\leq O\left(n\Delta log(n\Delta) + \sum_{L\in Labels[v]} \sum_{(v,u)\in A} \Delta log(n\Delta)\right)$$

$$\leq O\left(n\Delta log(n\Delta) + m\Delta^2 log(n\Delta)\right)$$

$$\leq O(m\Delta^2 log(n\Delta)) \tag{3.9}$$

□

3.5 Multi-destination Bi-objective Shortest Path Problem with Time windows

In real applications of vehicle routing problems, additional characteristics or constraints are considered to address the different specificities. One of the well-known constraints is customer time windows which ensures that the service of each customer starts within its associated time interval. In this section we explore how to consider the customer time windows when computing the multigraph representation for the road network and how to exploit these constraints in order to improve the performance of our algorithm.

Let e_i and l_i denote respectively the earliest starting service time and the latest starting service time for a customer i. A path $p_{v_iv_j}$ linking two customers v_i and v_j is feasible with respect to time windows if and only if $e_{v_i} + t(p_{v_iv_j}) \leq l_{v_j}$. Therefore, only paths satisfying this condition should be considered when constructing the multigraph representation for the road network.

We denote by $N^+(v_0)$ the subset of customers reachable from the source node v_0 within their time windows, *i.e.*, $N^+(v_0) = \{v_i, i = 1, ..., n_c; e_{v_0} + t^{inf}(v_0, v_i) \leq l_{v_i}\} \subset C$. Only destination nodes in $N^+(v_0)$ should be considered during the search procedure. To do this, we propose the following enhancements:

1. The detour in distance of a subpath associated with a label $L = (u, d(L), t(L))$ is evaluated regarding only reachable destination nodes. The key of the label L is reformulated and is given by $k(L) = \min_{v_i\in N^+(v_0)} (d(L) + d^{inf}(u, v_i) - d^{inf}(v_0, v_i))$;

2. The algorithm should terminate once labels $L_i = (v_i, d^{sup}(v_0, v_i), t^{inf}(v_0, v_i))$ have been generated only for destination nodes $v_i \in N^+(v_0)$. Thus, the stopping criterion is that the key of selected label is larger than $\max_{v_i\in N^+(v_0)} K_i^{sup}$ with $K_i^{sup} = d^{sup}(v_0, v_i) - d^{inf}(v_0, v_i)$;

3. Only nodes that are apt to lead to a feasible path to a destination node should be considered. Every node u that verifies $e_{v_0} + t^{inf}(v_0, u) + t^{inf}(u, v_i) > l_{v_i}$ for all destination nodes $v_i \in N^+(v_0)$ is discarded from the graph G.

Due to the first enhancement, the key values increase more quickly during the labeling procedure as $k(L) = \min_{v_i \in N^+(v_0) \subseteq C}(d(L) + d^{inf}(u, v_i) - d^{inf}(v_0, v_i)) \geq \min_{v_i \in C}(d(L) + d^{inf}(u, v_i) - d^{inf}(v_0, v_i))$ for every label L. The second enhancement aim to tighten the upper bound on the key values and to improve the stopping condition. The third enhancement is performed in preprocessing and permits to reduce the size of the considered network by removing "*useless*" nodes.

3.6 Computational experiments

In this section, we present the computational experiments carried out to evaluate the efficiency of the proposed solution method. First, we present the benchmark problems used in the experiments. Then, we report the computational results and we analyse the impact of considering the time windows on the algorithm performance.

3.6.1 Test problems

Since we are interested in computing paths for transportation problems, we conduct our computational experiments on the basis of three real-world road networks:

- Two road networks (*Aix-1* and *Aix-2*) are constructed based on spatial data from the city of *Aix-en-Provence* [1] in France provided by **OpenStreetMap©** [2] database. Each road segment is defined by a length, a maximum allowed speed and a travel direction. Travel times are then computed using speeds and lengths;
- The Road network (*DC*) of *Washington, D.C.*, United States, was extracted by Schultes [120] from **US Census** [3]. Each road segment is given with a distance and a travel time. Note that, the road network (DC) in the original data is undirected and we converted it into a directed network by duplicating all arcs.

Table 3.1 reports the main characteristics of the considered road networks. For each road network, the first two columns indicate respectively the number of nodes n and the number of

[1] Aix-en-Provence is a city-commune in the region of *Provence-Alpes-Cote d'Azur* in the south of France, about 30 km north of *Marseilles*

[2] OpenStreetMap is a collaborative project wich creates and distributes freely available geospatial data. `www.openstreetmap.org/`

[3] US Census 2000 TIGER/Line Files. U.S. Census Bureau, Washington, DC, Geography Division. `http://www.census.gov/geo/www/tiger/tigerua/uatgr2k.html`

Table 3.1: *Road networks characteristics*

			Outgoing arcs	
Name	# nodes	# arcs	min	max
AIX-1	5437	10098	1	4
AIX-2	19500	36203	1	5
DC	9559	29707	1	6

Table 3.2: *The number of tested instances for each road network*

	Number of destination nodes				
Network	$n_c = 25$	$n_c = 50$	$n_c = 100$	$n_c = 200$	$n_c = 500$
Aix-1	5	5	5	5	5
Aix-2	5	5	5	5	5
DC	5	5	5	5	5

arcs m. The last two columns indicate respectively the minimum and the maximum number of outgoing arcs over all nodes in the road network.

For each road network, we generate different instances with different number of destination nodes n_c. Table 3.2 details the number of tested instances for each road network and for each value of n_c. In each instance, $n_c + 1$ nodes were selected randomly to represent the set of key locations $C = \{v_0, v_1, ..., v_{n_c}\}$.

3.6.2 Results

In order to evaluate the performance of our algorithm, we compare the obtained results to those obtained using the basic label setting (LSET) and the label correcting (LCOR) algorithms. Recall that in the LSET algorithm, all non-dominated labels are sorted in a lexicographic order and at each iteration one label at a known node is set as permanent then is extended to successor nodes. In the LCOR algorithm, non dominated labels arriving at the same node are sorted in a lexicographic order, all labels are temporarily maintained during the search and become permanent at the last iteration.

All algorithms are implemented in the C++ programming language and tests are run on an Intel Xeon(R) CPU E5-2620v2 2.1 GHz computer with 32GB of memory.

In the following, we present first the results for the basic version of the multi-A* algorithm (without the enhancements proposed in Section 3.5), then we show the impact of the

Table 3.3: *Results for the road network Aix-1*

n_c		multi-A*	LSET	LCOR	# paths
25	1	14,8	51,1	22,7	100
	2	15,3	58,5	22,2	95
	3	16,8	56,4	22,8	109
	4	17,2	67,1	23,5	123
	5	16,7	55,2	24,3	106
50	1	19,5	58,2	23,9	233
	2	20,3	63,1	26,4	234
	3	18,2	51,3	21,9	216
	4	17,9	51,3	19,8	203
	5	18,5	54,6	21,8	208
100	1	21,5	51,9	22,2	413
	2	20,8	52,9	22,4	427
	3	20,5	51,2	21,6	421
	4	20,2	54,4	21,6	428
	5	20,3	52,9	22,9	427
200	1	23,7	57,1	23,3	890
	2	24,6	56,5	24,4	842
	3	23,5	56,4	22,6	835
	4	23,4	51,7	21,5	790
	5	22,6	49,2	21,8	782
500	1	30,3	53,6	20,8	2078
	2	29,9	51,7	21,4	2055
	3	30,4	56,2	25,3	2099
	4	30,2	53,6	23,2	2085
	5	30,2	53,9	22,8	2073

Table 3.4: *Results for the road network Aix-2*

n_c		multi-A*	LSET	LCOR	# paths
25	1	256,4	5301,2	832,2	301
	2	199,1	3021,9	531,7	278
	3	202,0	4250,4	670,1	288
	4	213,2	4459,4	632,2	292
	5	259,5	5122,3	825,2	314
50	1	375,5	6354,8	942,7	748
	2	278,6	4419,5	718,0	612
	3	329,2	4224,5	727,7	681
	4	213,0	4439,9	697,8	620
	5	301,0	4796,9	795,8	638
100	1	284,2	3686,9	748,2	1194
	2	253,6	3219,2	585,4	1070
	3	327,3	4785,4	756,6	1386
	4	331,1	4400,0	755,5	1302
	5	322,5	3590,0	707,9	1195
200	1	337,6	4529,0	708,7	2421
	2	377,2	4140,4	733,9	2585
	3	379,6	4230,3	681,4	2602
	4	413,6	4935,4	690,2	2730
	5	385,5	4094,8	639,9	2529
500	1	448,6	4740,3	661,5	6531
	2	439,4	4454,7	651,5	6470
	3	461,9	4457,5	685,9	6832
	4	447,6	4415,2	638,0	6467
	5	439,2	4357,9	686,6	6526

proposed modifications to take into account the customer time windows.

For each instance, all algorithms are applied $n_c + 1$ times where each time a node in $C = \{v_0, ..., v_{n_c}\}$ is selected to be the source node. Then, average computing times and the average number of Pareto-optimal paths for one-to-n case are reported. Obtained results are presented in Tables 3.3 and 3.4 for the road networks of the city of *Aix-en-Provence* and in Table 3.5 for *Washington D.C.* road network. Columns "multi-A*", "LSET" and "LCOR" report computing times (in milliseconds), respectively, for the multi-A*, LSET and LCOR algorithms. The average number of Pareto-optimal paths is presented in column "# paths".

From Tables 3.3 to 3.5, it comes out that the multi-A* algorithm performs quite well for real road networks. It provides Pareto-optimal paths with smaller computing times than the LSET algorithm for all instances in real road networks and it improves the computing times for all instances in Aix-2 and DC road networks and for 15 out of 25 instances in Aix-1 road network compared to the LCOR algorithm. The computing time with the multi-A* algorithm does not exceed 462 milliseconds for all instances in real road networks while it is up to 6577.8 milliseconds with the LSET algorithm and is up to 1171.6 milliseconds with the

Table 3.5: *Results for the road network DC*

n_c		multi-A*	LSET	LCOR	# paths
25	1	253,6	6469,9	896,2	483
	2	223,1	5475,0	886,0	411
	3	184,7	4311,4	768,2	328
	4	233,6	5526,6	1067,1	419
	5	254,3	5477,5	989,6	388
50	1	230,8	4328,5	1027,3	744
	2	243,9	4921,9	1107,4	792
	3	255,4	6254,2	944,3	899
	4	265,3	6102,9	902,1	896
	5	257,9	4992,3	811,2	784
100	1	301,0	5122,1	933,0	1583
	2	256,1	4859,5	999,2	1515
	3	298,7	6577,8	1056,1	1688
	4	290,6	5157,1	1107,8	1602
	5	245,7	4880,5	879,8	1371
200	1	311,7	5594,3	878,0	3149
	2	288,3	4486,7	767,8	2947
	3	294,3	4620,4	942,8	3028
	4	321,4	5651,6	940,3	3332
	5	283,4	4441,5	1078,4	2988
500	1	325,0	5062,2	1171,6	7917
	2	320,2	4741,9	1001,9	7645
	3	318,3	4738,0	978,3	7608
	4	321,4	4960,4	914,7	7819
	5	320,9	5066,8	969,7	7855

LCOR algorithm. The average speedup factor obtained with the multi-A* algorithm is up to 2.6 for instances in Aix-1, 14.1 for instances in Aix-2 and 19.2 for instances in DC road network compared to the LSET algorithm, and is up to 1.1 for instances in Aix-1, 2.3 for instances in Aix-2 and 3.5 for instances in DC road network compared to the LCOR algorithm.

Numerical results show that, for instances based on the same road network, extending the set of destination nodes increases significantly the computing time for the multi-A^* algorithm, while, computing times are almost constant with LCOR and LSET algorithms. For example, the average computing time for instances with $n_c = 25$ on *Aix-1* road network is about 16 milliseconds and reaches 30.3 milliseconds for instances with $n_c = 500$. On the other hand, average computing times for LSET and LCOR algorithms remain constant when going from $n_c = 25$ to $n_c = 500$: almost 23 milliseconds with the LCOR algorithm and about 55 milliseconds with the LSET algorithm. This is due to the fact that using basic labeling algorithms all labels over all the network have to be processed, while using the proposed goal-directed search strategy, the algorithm terminates once all Pareto-optimal paths at destination nodes have been generated. Therefore, increasing the number of destination could soften the stopping criteria and could enlarge the set of labels to be processed.

The impact of considering the customer time windows

To evaluate the impact of the proposed enhancements, we generate for each instance a set of time windows as follows: 1) A node $v_i \in C$ is, randomly, selected to represent the depot that defines the time horizon. 2) The time window $[0, T]$ associated with the depot are defined such that every node in C can visited on a route that starts and ends at the depot within the the time horizon T. 3) A set of routes are then constructed in a greedy way, so that every customer is visited exactly by one route. 4) Finally, the time windows $[e_{v_i}, l_{v_i}]$ are defined such that the constructed routes are feasible and such that $l_{v_i} = e_{v_i} + \frac{T}{5}$.

Note that, in the following we focus on results obtained with the muli-A* algorithm and we do not report those obtained using LCOR and LSET algorithms taking into account the time windows. This is because that considering time windows on source and destination nodes does not significantly impact the structure of the LCOR and the LSET algorithms: the only possible modification is to check if the travel time associated with obtained label at every extension does not exceed a precomputed upper bound given by $\underset{1 \leq i \leq n}{max}(l_{v_i} - e_{v_0})$. Consequently, the performance of these algorithm would not be extremely improved.

Tables 3.6 to 3.8 present some statistics on the impact of considering the time windows on the preprocessing phase. In these tables, the average number of reachable destinations and the average number of removed nodes for a given source node are reported, respectively, in columns "$|N^+(v_0)|$" and "# Removed nodes". The last two columns report respectively preprocessing times in the basic multi-A* algorithm and in the multi-A* algorithm with the

Table 3.6: *Impact of considering Time Windows on the preprocessing with the instances on Aix-1*

			Preprocessing CPU	
n_c	$\|N^+(v_0)\|$	# Removed nodes	without TW (ms)	with TW (ms)
25	13	703	9,0	9,1
50	28	522	9,5	9,6
100	54	678	10,5	10,8
200	116	403	12,4	13,5
500	294	226	18,4	22,4

Table 3.7: *Impact of considering Time Windows on the preprocessing with the instances on Aix-2*

			Preprocessing CPU	
n_c	$\|N^+(v_0)\|$	# Removed nodes	without TW (ms)	with TW (ms)
25	13	2881	35,5	35,4
50	26	3766	36,7	36,5
100	51	3373	40,4	40,9
200	101	3311	48,5	49,0
500	255	2902	69,2	77,0

proposed enhancements.

From Tables 3.6 to 3.8, we notice that, by considering the time windows, the number of considered destinations $|N^+(v_0)|$ for a source node v_0 is significantly reduced: in average only 53% of the destination nodes are reachable from the source node v_0 within their time windows. Therefore, a lot of computational efforts could be saved by focusing only on nodes $v_i \in N^+(v_0)$ during the search procedure. We also see that the number of "*useless*" nodes, *i.e.*, nodes v for which there is no visible paths from v_0 to v_i through v for all $v_i \in N^+(v_0)$ is relatively important. Removing these nodes from the considered network could speed up the search scheme. From columns "Preprocessing without TW (ms)" and "Preprocessing with TW (ms)", we observe that the proposed enhancements increases slightly the preprocessing times. This increase does not exceed 8 milliseconds (for instances with 500 destinations on *Aix-2* road network). In addition, this increase gets larger when the number of destination nodes increases.

In Tables 3.9 to 3.11 we compare the results obtained with the basic multi-A* algorithm

Table 3.8: *Impact of considering Time Windows on the preprocessing with the instances on DC*

			Preprocessing CPU	
n_c	$\|N^+(v_0)\|$	# Removed nodes	without TW (ms)	with TW (ms)
25	14	1245	23,2	23,4
50	27	1778	24,1	24,9
100	51	1821	26,0	25,7
200	95	1984	29,7	29,1
500	239	2000	40,1	41,6

Table 3.9: *Impact of considering Time Windows for instances on Aix-1*

		multi-A* without TW			multi-A* with TW		
n_c		labeling (ms)	CPU(ms)	# paths	labeling (ms)	CPU(ms)	# paths
25	1	5,9	14,8	100	3,7	12,7	46
	2	6,2	15,3	95	2,7	11,9	46
	3	7,8	16,8	109	4,6	13,7	46
	4	8,0	17,2	123	4,6	13,8	63
	5	7,7	16,7	106	4,2	13,3	45
50	1	9,9	19,5	233	6,1	15,7	96
	2	10,7	20,3	234	7,4	17,1	126
	3	8,9	18,2	216	6,5	16,1	105
	4	8,5	17,9	203	6,1	15,8	108
	5	9,0	18,5	208	6,1	15,5	111
100	1	10,9	21,5	413	8,1	18,8	198
	2	10,3	20,8	427	7,3	18,1	195
	3	10,0	20,5	421	6,5	17,1	170
	4	9,8	20,2	428	7,1	17,9	210
	5	10,0	20,3	427	7,2	18,0	226
200	1	11,3	23,7	890	8,8	22,3	470
	2	12,0	24,6	842	9,3	22,9	469
	3	11,2	23,5	835	8,9	22,3	454
	4	11,0	23,4	790	8,8	22,3	422
	5	10,3	22,6	782	7,6	21,0	381
500	1	11,8	30,3	2078	10,5	33,0	1210
	2	11,4	29,9	2055	9,8	32,3	1093
	3	11,9	30,4	2099	9,4	31,7	966
	4	11,7	30,2	2085	10,8	33,2	1263
	5	12,0	30,2	2073	10,2	32,5	1152

(reported in column "multi-A* without TW") and the results obtained with the enhanced multi-A* algorithm (reported win column "multi-A* with TW"). For each algorithm, column "labeling (ms)" indicates the average computing time (in milliseconds) for the labeling procedure, column "CPU(ms)" indicates the total computing time (in milliseconds) including preprocessing and labeling procedure execution times, and column "# paths" indicates the average number of Pareto optimal paths. Recall that reported results are expressed in averages for one source node: the complete multigraphs are first constructed by applying $|n_c|$ times each algorithm with a source node $s \in \{v_0, ..., v_{n_c}\}$, then the average computing times and the average number of paths for a single source are reported.

From tables 3.9 to 3.11, it comes out that, using the introduced enhancements based on customer time windows, the labeling procedure becomes faster. The computing time for the labeling procedure is reduced for all instances. The average improvements on computing time for the labeling are 2.7, 124.7 and 100.2 milliseconds respectively for instances on *Aix-1*, *Aix-2* and *DC* road networks. Consequently, for instances with low increase in the preprocessing time, the total computing time is significantly reduced. We notice that except for instances with $n_c = 500$ on *Aix-1* road network, the total computing time is improved for

Table 3.10: *Impact of considering Time Windows for instances on Aix-2*

		multi-A* without TW			multi-A* with TW		
n_c		labeling (ms)	CPU(ms)	# paths	labeling (ms)	CPU(ms)	# paths
25	1	220,0	256,4	301	70,7	106,2	113
	2	163,6	199,1	278	76,7	111,4	99
	3	166,7	202,0	288	65,3	100,8	97
	4	178,7	213,2	292	89,9	125,4	125
	5	223,8	259,5	314	81,7	117,2	95
50	1	339,2	375,5	748	164,9	202,6	273
	2	242,2	278,6	612	107,0	142,6	212
	3	292,1	329,2	681	149,8	186,1	275
	4	176,7	213,0	620	81,2	117,7	234
	5	263,5	301,0	638	194,4	230,5	292
100	1	244,3	284,2	1194	144,0	185,4	450
	2	213,5	253,6	1070	106,6	147,1	374
	3	287,1	327,3	1386	132,4	172,8	436
	4	290,2	331,1	1302	209,9	251,9	592
	5	282,1	322,5	1195	149,5	189,8	422
200	1	289,2	337,6	2421	202,1	253,1	1091
	2	328,8	377,2	2585	196,3	245,2	886
	3	331,1	379,6	2602	176,6	226,0	989
	4	365,1	413,6	2730	208,0	256,9	1010
	5	336,7	385,5	2529	139,0	186,0	700
500	1	379,2	448,6	6531	273,2	351,2	2642
	2	370,4	439,4	6470	266,9	344,3	2549
	3	392,5	461,9	6832	269,5	348,4	2938
	4	378,2	447,6	6467	251,6	329,0	2533
	5	370,3	439,2	6526	201,5	274,6	1866

Table 3.11: *Impact of considering Time Windows for instances on DC*

		multi-A* without TW			multi-A* with TW		
n_c		labeling (ms)	CPU(ms)	# paths	labeling (ms)	CPU(ms)	# paths
25	1	230,3	253,6	483	178,2	201,8	250
	2	199,9	223,1	411	129,4	152,5	216
	3	161,6	184,7	328	92,0	116,0	171
	4	210,0	233,6	419	115,8	139,3	174
	5	231,6	254,3	388	148,0	170,8	147
50	1	206,6	230,8	744	121,8	146,6	402
	2	219,8	243,9	792	116,9	141,3	330
	3	231,3	255,4	899	127,1	151,7	431
	4	241,3	265,3	896	150,8	175,8	456
	5	233,9	257,9	784	88,4	114,1	269
100	1	275,7	301,0	1583	164,5	190,3	728
	2	230,0	256,1	1515	149,4	175,3	701
	3	272,9	298,7	1688	187,0	213,0	793
	4	264,9	290,6	1602	137,1	162,6	635
	5	219,1	245,7	1371	101,1	126,4	511
200	1	282,1	311,7	3149	202,7	231,9	1467
	2	258,4	288,3	2947	157,5	186,6	1169
	3	264,7	294,3	3028	176,8	206,9	1458
	4	291,9	321,4	3332	130,9	159,0	1016
	5	253,7	283,4	2988	125,1	153,8	938
500	1	285,1	325,0	7917	161,0	201,2	2612
	2	280,3	320,2	7645	163,5	204,3	2679
	3	278,2	318,3	7608	205,2	248,6	3603
	4	281,1	321,4	7819	136,1	175,3	2278
	5	280,9	320,9	7855	215,0	259,5	4086

all instances. Computing times are in average reduced by 9%, 39% and 36% respectively for instances on *Aix-1*, *Aix-2* and *DC* road networks. These savings reach 3.8 milliseconds (for instance 1 with $n_c = 50$, table 3.9), 199.5 milliseconds (for instance 5 with $n_c = 200$, table 3.10) and 162.4 milliseconds (for instance 4 with $n_c = 200$, table 3.11). We also observe that the reduction in computing time decreases when the number of the destination nodes increases, *e.g.*, computing time is reduced in average by 59% for instances with $n_c = 25$ on *Aix-2* road networks and is reduced in average by 26.4% for instances with $n_c = 500$ on the same road network. Note that for instances with 500 destination nodes on *Aix-1*, in spite the savings in computing time for the labeling procedure (1.7 milliseconds in average), the total computing time is more important for the multi-A* algorithm with the proposed enhancements (32.6 milliseconds in average compared to 30.2 milliseconds in average with the basic multi-A* algorithm). This increase in the total computing time is due to the increase in the preprocessing time (4.0 milliseconds) that is more important than the savings in time for the labeling procedure.

3.7 Conclusions

In this paper, we investigated the multi-destination bi-objective shortest path problem. The particularity of this problem is to seek for all Pareto-optimal paths linking a source node to a subset of nodes in a road network. We developed an exact solution method based on a labeling approach combined with a modified A^* algorithm. The proposed algorithm is based on a goal-directed search strategy that permits to guide the labeling procedure simultaneously towards all destination nodes and to terminate the search once all optimal solutions have been determined. We also proposed some enhancements for the algorithm based on time windows defined for points of interest.

Computational experiments were carried out on a large panel of instances based on real road networks. Results show that the proposed algorithm performs very well for all tested instances. Compared to basic labeling approaches, the multi-A* algorithm outperforms the label setting algorithm for all instances and improves computing times with the label correcting algorithm for most instances with large number of nodes in the road networks. Finally, we illustrated the impact of considering time windows and the significant savings on computational efforts that can be obtained by adapting the algorithm to take into account these time windows.

Chapter 4

Empirical analysis for the VRPTW with a multigraph representation for the road network

This chapter is an accepted manuscript in *Computers & Operations Research* journal (doi: 10.1016j.cor.2017.06.024).

Abstract

Vehicle routing problems have drawn researchers' attention for more than fifty years. Most approaches found in the literature are based on the key assumption that for each pair of points of interest (e.g., customers, depot...), the best origin-destination path can be computed. Thus, the problem can be addressed via a complete graph representation, so-called customer-based graph, where nodes represent points of interest and arcs represent the best paths. Yet, in practice, it is common that several attributes are defined on road segments. Consequently, alternative paths presenting different trade-offs exist between points of interest. In this study, we investigate in depth a special representation of the road network proposed in the literature and called a multigraph. This representation enables one to maintain all these alternative paths in the solution space. We present an empirical analyses based on the Vehicle Routing Problem with Time Windows, as a test bed problem, solved with branch-and-price algorithms developed for the different types of graphs. Computational experiments on modified benchmarks from the literature and on instances derived from real data evaluate the impact of the modeling on solution quality.

Keywords: Vehicle Routing Problems, Road networks, Multigraph, Branch-and-Price.

4.1 Introduction

The vehicle routing problem is one of the most extensively studied classes of combinatorial optimization problems in the operational research literature. One reason is the large number and the interest of its applications in logistics, supply chain management, distribution systems, car navigation systems, etc. Although this research area has been broadly explored in the last fifty years, most works are built on an assumption that is at best disputable, and at worst can lead to a bad optimization of vehicle routes.

A vehicle routing problem aims at planning a set of routes on a given road network, so as to cover a set of customer requests with a fleet of vehicles. Most models proposed in the literature represent the road network with a weighted complete graph. Nodes are introduced for the different points of interest (e.g., customers, depot...), and arcs correspond to shortest paths computed according to a single criterion, generally travelling cost, distance or travelling time, between these points of interest.

In real-life applications and especially in urban areas, several operational constraints are implied and objectives of the involved partners must be taken into account. In many cases, different attributes have to be defined for each road segment in the original road network. Hence, each pair of nodes may be connected with a set of paths proposing different compromises between the considered attributes. In such situations, representing the problem with a customer-based graph, i.e., with only one arc between each pair of nodes, could discard many potentially good solutions from the solution space. To handle this issue, an alternative modeling approach, so-called multigraph representation, was proposed by Garaix et al. in [61]. This representation aims at considering all non-dominated paths linking each two points of interest in the solution space.

A typical example is provided by the Vehicle Routing Problem with Time Windows (VRPTW). In this problem, transportation plans are constrained to satisfy costumer requests within their time windows. Each road segment is defined with a cost and a travelling time. In the standard setting, the problem is defined on a complete graph and each arc represents a best path. However, the cheapest path is unlikely to be the same as the fastest path due to, for example, heavy traffic and congestion in some short road segments or additional charges for high-speed routes. When defining its transportation plan, a carrier might prefer an expensive road segment in case of hard time constraints or, conversely, a cheapest one when time constraints are soft. Consequently, representing the problem with a weighted customer-based graph could lead to operational solutions with an overestimated cost, or, even worse, to the false conclusion that no feasible solution exists. To illustrate this, let us consider the small road network provided on **Figure 4.1**.

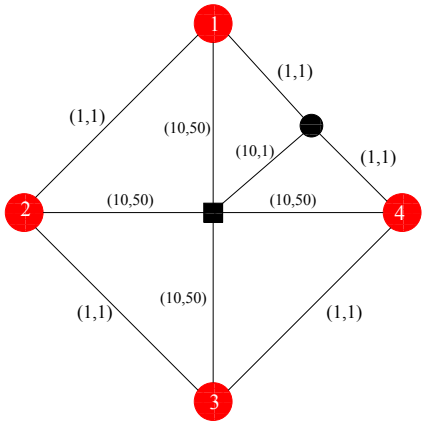


Figure 4.1: *Illustrative road network*

In this example, the depot is located at node 0 (represented by the black square), and customers are located at nodes 1, 2, 3 and 4. Each edge represents a road segment. The black circle node corresponds to a junction of three road segments. For each edge, the travelling cost and time are provided in parentheses, in this order *(cost, time)*. The objective is to determine a set of vehicles routes that visit all customers and return to the depot within 100 units of time.

If we consider the customer-based graph where arcs represent cheapest paths (**Figure 4.2.a**), the best solution costs 80 and consists in visiting each customer with a different vehicle. Considering all alternative paths, a multigraph representation (**Figure 4.2.b**) allows one to obtain a better solution that serves all customers with the same vehicle with a total travelling cost equal to 24. Note furthermore that for a lower time-limit (or for a limited fleet), no feasible solution could have been found with the customer-based graph. Note also that equivalent examples could be constructed with a customer-based graph obtained from fastest paths, except that in this case a feasible solution would always be found when it exists.

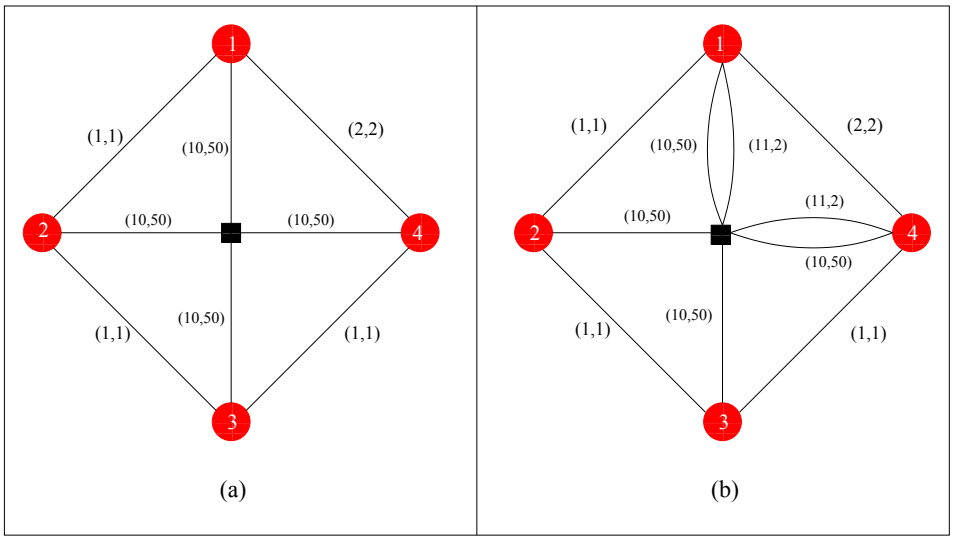


Figure 4.2: *Complete graph and multigraph representations*

Following the above remarks, we investigate in this work two important issues:

1. Is it tractable to represent the road network with a multigraph, where, for each pair of nodes, the set of alternative arcs is the set of non-dominated Pareto optimal paths computed according to road segment attributes?
2. Could this representation have a significant impact on solution costs in practice?

The first issue relates to the computation and size of the multigraph, and to the efficiency of solution methods. The second issue relates to experimental analyses and comparisons between customer-based graph and multigraph representations. For our investigations, we use the VRPTW as a test-bed problem.

Besides the case of the VRPTW, where cost and time are associated with arcs, many other attributes might be considered in vehicle routing depending on the context. Greenhouse gas emissions (that depend on travel distances but also average speeds, slopes...) would be considered by a decision-maker who attempts to limit environmental impacts when defining its transportation plans. Energy consumption (that also depends on distances, speeds, slopes...) has to be taken into account when electric vehicles are operated. An attribute modeling scenic beauty could be introduced when optimizing sightseeing tours. Safety (for hazmat or cash transportation), transport mode, robustness define other examples.

The remainder of the paper is organized as follows. In Section 4.2, we review the relevant literature and give more insights into the methodology followed in this work. In Section 4.3, we present a mathematical formulation for the VRPTW with the multigraph representation. We describe, in Section 4.4, a branch-and-price solution approach adapted to the multigraph setting. Finally, in Section 4.5 we present the results of extensive experiments conducted to evaluate the impact of the modeling approach and the efficiency of our method.

4.2 Literature review

Vehicle Routing Problems are widely studied and a large number of solution approaches are proposed in the literature. The large majority of these approaches are based on the key assumption that one can compute the best path for all pairs of nodes. Thus, the problem can be tackled using a customer-based graph. As mentioned before, this assumption is not guaranteed to hold when several attributes are defined on arcs. In the literature, few works evoke this issue. In this section, we overview these works. In some of them, a multigraph representation is investigated. In others, the problem is directly solved in the original road network. These two options are surveyed in the two next subsections, respectively. A subsequent subsection

details how our work intends to complete this literature and motivates the methodology that we proposed to follow.

4.2.1 Multigraph representation

The first possibility consists in representing the road network with a multigraph, so that alternative routes are considered between each pair of points of interest.

A key paper in this regard was proposed by Garaix et al. [61]. As far as we know, they were the first to point out that when several attributes are defined on arcs, one cannot transform a vehicle routing problem on a road network into a standard VRP without taking the risk of losing optimality, and to explore this issue. Their motivation stemmed from the development of a real-world *On-Demand Transportation* system. For this reason, they were guided by the objective of efficiently solving a specific Dial-a-Ride Problem. They introduced a multigraph representation and developed two specialized solution approaches: a simple insertion heuristic and a branch-and-price procedure. Experiments were mainly designed to evaluate practical objectives, but they also demonstrated that important improvements could be obtained using the multigraph representation compared to a fastest-path-based complete graph.

Lai et al. [85] considered this issue for the heterogeneous VRP with limited duration. Following Garaix et al. [61], they introduced alternative arcs between pair of vertices. They proposed a tabu search and insisted on how neighborhood exploration should be modified to consider the multigraph structure. Experiments were carried out on randomly generated instances with two alternative arcs between every vertex pair. The advantages of introducing the alternative arcs were largely investigated. The experiments confirmed, on a different problem, the observations made in Garaix et al. [61].

Besides the two aforementioned papers that explicitly investigate the limits of the customer-based graph, several other papers also consider multigraphs, with relatively similar goals.

Wang and Lee [133] introduced the so-called Time Dependent Alternative Vehicle Routing Problem (TDAVRP) that also involves a multigraph representation. The TDAVRP is a vehicle routing problem with time windows and travel times depending on the time of the day. Each pair of nodes is connected with two edges. The first one is called the designated edge and is assigned a time-dependent travel speed distribution. The other one represents an alternative route with a constant travel time. This route is longest but is not sensitive to traffic and could be used during peak hours. To solve this problem, Wang and Lee [133] developed a heuristic algorithm based on Particle Swarm Optimization (PSO).

Caramia and Guerriero [17] studied a long-haul freight transportation problem motivated by a real-life application implying multimodal routes. Travel time and route cost are to be minimized together with the maximization of a transportation-mean sharing index. They observed that the multimodality in transportation problems enlarges significantly the set of possible solutions and makes the definition of an optimum respecting the numerous objectives very difficult. In this study, the transportation network contains alternative routes, thus generating a multimodal multigraph: some arcs link the same pair of nodes with the same transportation mode but with different costs and traveling times, some arcs have the same endpoints but are associated with different modes. To solve this problem, the authors implemented a heuristic algorithm based on local search.

More recently, Reinhardt et al. [113] introduced a new generalization of the VRPTW in which additional fixed costs are associated with subsets of edges. They noticed that, in some real-life situations, fees must be paid for accessing roads, areas or bridges. These fees could reflect payment for toll roads, ferry connections, investment in new facilities and the need for certifications to access to war zones or areas of unrest. Then, connection costs between customer locations do not depend only on traveling costs, but also rely on other decisions, basically buying or not access to a set of routes in the transportation network. Hence, a standard VRPTW does not model correctly the problem. Reinhardt et al. [113] proposed an extension of the VRPTW to represent the problem, called *Edge Set Vehicle Routing Problem with Time Windows*. In this new problem, edges are regrouped in different subsets, each subset being characterized by a fixed cost. Once this cost is paid, all vehicles can access all edges in the associated subset. The problem is modeled by a multigraph, in order to differentiate between edges that connect the same pair of nodes but do not belong to the same edge subset. In their solution method (branch-price-and-cut) and experiments, the authors however only address the special case when the graph is simple, *i.e.*, only one edge exists between each pair of nodes). The multigraph case is left as a perspective.

4.2.2 Road Network

Recently, Letchford et al. [89] revisited the branch-and-price approach presented in [61] (see Subsection 4.2.1), for the VRPTW. Their objective was to show that it was more efficient to model and solve the problem directly on the road network than to introduce the multigraph. They explained how it would impact both the pricing problem and the branching scheme, but only explored the pricing problem. In their experiments, they demonstrated the interest of this approach. The complete algorithm, including branching schemes, was left for future researches.

It is worth mentioning the connections between [89] and a stream of papers addressing variants of the Travelling Salesman Problem (TSP) defined on general graphs, such as road networks. An early work was carried out by Fleischmann [56]. Its motivation was the so-

lution of the Traveling Salesman Problem (TSP) with a cutting-plane approach. Instead of searching for the min-cost Hamiltonian cycle in a complete graph, he proposed to search for a cycle visiting all required nodes in the underlying road network. He called this problem the TSP on a Road Network or R-TSP. Independently, Cornuéjols et al. [29] investigated the same problem that they called the Steiner Traveling Salesman Problem (STSP). The rationale behind Fleischmann's suggestion was that nodes in road networks typically have small degrees: adding artificial arcs to complete the original graph increases enormously the number of variables needed in the linear programming formulation. In the same paper, the author also suggested a way to extend the solution procedure to the VRP, but revealed the difficulties that would arise in this case. Recently, in connection with [89], Letchford et al. [90] also addressed the STSP. A more general problem was introduced by Orloff [103] in which services are associated with both a subset of nodes and a subset of arcs in the road network.

Finally, Huang et al. [70] investigated a multi-objective TSP in the context of a tourist sightseeing-itinerary planning. Since the tourist had to visit a known subset of nodes in a road network, the problem could be addressed as a variant of the TSP. Each road segment was defined with four attributes related to four objectives: travel time, vehicle operating cost, safety level and surrounding scenic view quality. The authors however proposed to optimize a weighted sum of the four objectives. Then, since no constraints were defined on the attributes, it made an aggregation of the four attributes possible, which allowed the authors to compute best-paths between required nodes and classically transform the road network into a complete graph.

4.2.3 Methodology

This literature review confirms that the traditional customer-based graph used in vehicle routing can have a significant (negative) effect on the solution quality, when several attributes are defined on road network arcs. Garaix et al. [61] and Lai et al. [85] gave empirical evidences of these effects in two very different contexts.

In this paper we propose to conduct further analyses. Lai et al. [85] acknowledged several limits in their conclusions. Their instances were randomly generated and had only two alternative arcs between every vertex pair. In addition, their comparisons were based on results found with a heuristic. Garaix et al. [61] based their conclusions on exact solutions, but only compared the multigraph representation to the fastest-paths-based complete graph. In addition, while they experimented on a real road network, this road network was that of a rural area. Conclusions can then hardly be generalized to urban networks.

For all these reasons, we propose to complete these results by considering what is probably the simplest and most studied vehicle routing problem with two attributes: the VRPTW. Also, for the sake of completeness, we propose to base our experiments on several types of

instances: (1) Solomon's instances, that are a must-do for the VRPTW; (2) realistic instances constructed following Letchford et al. [89]; (3) instances obtained from real road networks. Finally, in order to be able to draw our conclusions from exact results and to have a fair and complete comparison, we propose to apply a branch-and-price methodology to the multigraph and to two customer-based graphs: the fastest-path-based and the cheapest-path-based.

Letchford et al. [89] suggested that a branch-and-price algorithm for the VRPTW that works directly on the original road network, rather than on a multigraph, could be more efficient in some cases. Unfortunately, although they showed how to perform pricing on the road network, they did not devise a suitable branching rule. For this reason, in this work, we only consider the multigraph representation. Further remarks on this issue can be found in Section 4.6.

4.3 Problem formulation

This section introduces the multigraph-based VRPTW, and proposes a mathematical formulation that generalizes a classical VRPTW model.

Let $G = (V, A)$ be a directed multigraph induced by a road network. $V = \{0, 1, \ldots, n\}$ is a set of nodes where the node 0 represents the central depot and $C = \{1, \ldots, n\}$ is the set of customers to serve. $A = \bigcup_{(i,j)\in V^2} A_{(i,j)}$ denotes the set of arcs where $A_{(i,j)} = \{(i, j)^p; p = 1, \ldots, |A_{(i,j)}|\}$ represents the set of alternative paths linking the two nodes i and j.

We associate with each arc $(i, j)^p$ a travel time $t_{(i,j)^p}$ and a cost $c_{(i,j)^p}$ that represent respectively the time needed and the cost induced to go from node i to node j through the associated path indexed by p (sequence of road segments) in the road network. With each customer i is associated a demand d_i, a time window $[e_i, l_i]$ and a service time s_i. To serve the set of customers, we use a set of homogeneous vehicles K with a capacity Q. The objective is to minimize the total cost.

Variables are defined as follows:

$x^k_{(i,j)^p}$: binary variable equal to 1 if arc $(i, j)^p$ is traversed by vehicle k and 0 otherwise.

t^k_i: starting time of the service at customer i if it is visited by vehicle k, meaningless value otherwise.

The mathematical model of the VRPTW on a multigraph is given in the following:

$$Min \sum_{k\in K} \sum_{(i,j)^p \in E} c_{(i,j)^p} x^k_{(i,j)^p} \tag{4.1}$$

$$s.t. \quad \sum_{k\in K} \sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} x^k_{(i,j)^p} = 1 \qquad i \in C \tag{4.2}$$

$$\sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} x^k_{(j,i)^p} - \sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} x^k_{(i,j)^p} = 0 \qquad i \in V, k \in K \tag{4.3}$$

$$\sum_{i\in V} \sum_{p=1}^{|A_{(0,i)}|} x^k_{(0,i)^p} \leq 1 \qquad k \in K \tag{4.4}$$

$$\sum_{i\in V} \sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} d_i x^k_{(i,j)^p} \leq Q \qquad k \in K \tag{4.5}$$

$$t^k_i + s_i + t_{(i,j)^p} x^k_{(i,j)^p} \leq t^k_j + M(1 - x^k_{(i,j)^p}) \qquad i, j \in V, 1 \leq p \leq |A_{(i,j)}, k \in K \tag{4.6}$$

$$e_i \leq t^k_i \leq l_i \qquad i \in V, k \in K \tag{4.7}$$

$$x^k_{(i,j)^p} \in \{0, 1\} \qquad i, j \in V, 1 \leq p \leq |A_{(i,j)}|, k \in K \tag{4.8}$$

$$t^k_i \geq 0 \qquad i \in V, k \in K \tag{4.9}$$

The objective function (4.1) minimizes the total travelling cost. Constraints (4.2) guarantee that each costumer is visited exactly once. Constraints (4.3) ensure that each vehicle arriving at a customer leaves to another. Constraints (4.4) ensure that vehicles do not start from the depot more than once. Inequalities (4.5) state that a vehicle can only be loaded up to its capacity. Inequalities (4.6) establish the relationship between the service starting times at a costumer and at its immediate successor. Moreover, inequalities (4.6) prevent sub-tours. Finally, constraints (4.7) ensure the respect of time windows and (4.8) are the integrity constraints. M is a large constant.

4.4 Solution Method

In this section, we present a solution method for the multigraph based VRPTW which is based on a branch-and-price procedure. The motivation for using a branch-and-price technique is that this solution approach has performed very well for many transportation problems and has become one of the most efficient exact methods to solve vehicle routing problems.

The branch-and-price has first been applied to the VRPTW by Desrochers et al. [40]. It is based on the Dantzig-Wolfe decomposition [32]. This decomposition gives rise to an integer master problem that has a tighter linear programming relaxation than the compact

formulation (4.1)-(4.9). Therefore, the master formulation is more suitable for a branch-and-bound scheme. However, this formulation involves a huge number of variables. To deal with these variables a column generation procedure is embedded in the branch-and-bound framework. The column generation sub-problem, so-called pricing problem, aims at finding a set of feasible columns that will be added to the master problem [39, 52, 37, 41]. In the following subsections, we briefly describe the different components of the branch-and-price framework and outline the main modifications that we made to handle the multigraph setting.

We emphasize that our objective in this paper is not to achieve the best possible implementation of the branch-and-price for the multigraph-based VRPTW. Instead, we aim at developing a method that allows deriving conclusive results.

4.4.1 Master Problem

Let Ω be the set of feasible vehicle routes, i.e., the set of paths in the multigraph starting from the depot, visiting a subset of customers respecting capacity and time windows constraints and returning to the depot. Let $a_{i,k} = 1$ if route $r_k \in \Omega$ visits customer i and $a_{i,k} = 0$ otherwise. The cost c_k of route r_k is given by $c_k = \sum_{i \in V} \sum_{j \in V} \sum_{p=1}^{|A_{(i,j)}|} \alpha^k_{(i,j)^p} c_{(i,j)^p}$ where $\alpha^k_{(i,j)^p} = 1$ if vehicle k uses the p^{th} arc to go from i to j and $\alpha^k_{(i,j)^p} = 0$ otherwise. Using this notation, we can express $a_{i,k}$ as $a_{i,k} = \sum_{j \in V} \sum_{p=1}^{|A_{(i,j)}|} \alpha^k_{(i,j)^p}$.

The multigraph based VRPTW can be formulated as a set covering problem :

$$Min \sum_{r_k \in \Omega} c_k z_k \tag{4.10}$$

$$s.t. \quad \sum_{r_k \in \Omega} a_{i,k} z_k \geq 1 \quad i \in C \tag{4.11}$$

$$\sum_{r_k \in \Omega} z_k \leq |K| \tag{4.12}$$

$$z_k \in \{0, 1\} \qquad r_k \in \Omega \tag{4.13}$$

Variable z_k is a binary variable such that $z_k = 1$ if route r_k is used in the solution and $z_k = 0$ otherwise. Constraints (4.11) ensure that each customer is visited at least once. Constraints (4.12) limit the number of used vehicles to the fleet size.

As in the case of the standard VRPTW, the set covering model cannot be solved using a standard branch-and-bound procedure. This is due to the exponentially growing size of set

Ω. This issue can be tackled by using a column generation technique and a branch-and-price scheme instead of branch-and-bound. In the following, we denote by Master Problem (MP) the linear relaxation of model (4.10)-(4.13).

4.4.2 Column Generation

Column generation is based on two components:

- A restriction of the master problem to a subset of routes $\Omega_t \subseteq \Omega$, denoted by MP($\Omega_t$).
- A pricing problem that generates and adds iteratively new columns to Ω_t.

The general scheme of the column generation is described by **Algorithm 4.1**.

Algorithm 4.1 Column generation algorithm

1: $t \leftarrow 0$
2: **repeat**
3: Solve the Master Problem restricted to set of routes Ω_t; MP(Ω_t)
4: $R \leftarrow$ Routes with negative reduced costs generated by the Pricing Problem
5: $\Omega_{t+1} \leftarrow \Omega_t \cup R$
6: $t \leftarrow t + 1$
7: **until** $R = \emptyset$

In our implementation, we start by initializing Ω_0 with a simple set of routes obtained using an adapted *Savings Algorithm*. At each iteration, we solve the restricted master problem MP(Ω_t) using a standard MILP solver. To enrich Ω_t, the pricing problem focuses on finding new routes offering better ways to visit customers, i.e., routes with negative reduced costs. Let $\lambda_i \geq 0$, $i \in C$, be the dual variables associated with Constraints (4.11) and $\lambda_0 \leq 0$ the dual variable associated with Constraint (4.12). The reduced cost $\hat{c}_k$ of route r_k is defined as follows :

$$\hat{c}_k = c_k - \sum_{i \in V} a_{i,k} \lambda_i \tag{4.14}$$

or equivalently,

$$\hat{c}_k = \sum_{i \in V} \sum_{j \in V} \sum_{p=1}^{|A_{(i,j)}|} \alpha^k_{(i,j)^p} (c_{(i,j)^p} - \lambda_i) \tag{4.15}$$

So, the pricing problem aims at generating routes $r_k \notin \Omega_t$ such that

$$\sum_{i \in V} \sum_{j \in V} \sum_{p=1}^{|A_{(i,j)}|} \alpha^k_{(i,j)^p} (c_{(i,j)^p} - \lambda_i) < 0 \tag{4.16}$$

As for the standard VRPTW, the pricing problem for the multigraph version can be reduced to an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC). Here, the ESPPRC consists in finding elementary paths in the multigraph starting and ending at the depot, satisfying capacity and time window constraints, and minimizing costs given by arc costs $c'_{(i,j)^p} = c_{(i,j)^p} - \lambda_i$. Note that the pricing problem can be reduced to a more tractable sub-problem which is the (non-elementary) Shortest Path Problem with Resource Constraints (SPPRC). Desrochers et al. [40] observed that the ESPPRC, thought NP-hard in the strong sense, admits a pseudo-polynomial algorithm when the elementary path condition is removed. With this version of pricing problem, slight modifications have to be made to the branch-and-price framework: Ω includes non elementary paths, $\alpha^k_{(i,j)^p}$ denotes the number of times the arc $(i, j)^p$ is traversed in route r_k and $a_{i,k}$ becomes the number of times the customer i is visited in route r_k. Although the state space of the pricing problem is larger in this case, the SPPRC can be solved more efficiently than the ESPPRC. However, with the SPPRC, the restricted master problem provides a weaker lower bound which complicates the pruning of nodes during the search tree. For this reason, we opt for the ESPPRC in the pricing. To solve this problem, we adapt the dynamic programming algorithm described in [53] to the multigraph case of the ESPPRC. Basically, the algorithm is an extension of Bellman-Ford algorithm and consists in associating with each partial path a label and extending these labels.

With the multigraph representation, more than one label can be obtained when extending a partial path to a customer j. A new path is obtained for each arc connecting the final node i of the current label to j as long as resource constraints allow it. We use dominance rules based on resource consumption to discard some of the labels arriving at the same location. At the end, the algorithm gives the best feasible paths.

Recall that when constructing the multigraph, we consider all non-dominated arcs. Due to this structure, we know that travel times associated with fastest arcs verify the triangle inequality, i.e.:

$$\forall (i, j, k) \in V^3 \text{ with } A_{(i,j)} \neq \emptyset, A_{(i,k)} \neq \emptyset \text{ and } A_{(j,k)} \neq \emptyset : t_{(i,j)^{|A_{(i,j)}|}} + t_{(j,k)^{|A_{(j,k)}|}} \geq t_{(i,k)^{|A_{(i,k)}|}} \quad (4.17)$$

assuming that arcs in sets $A_{(i,j)}$ are ranked from the cheapest to the fastest.

Using this property, we can detect and eliminate unreachable nodes for every partial path. Let L be a label and i the last node visited on the partial path represented by L. If a customer j cannot be reached by extending L along arc $(i, j)^{|A_{(i,j)}|}$, j is unreachable for all labels obtained by extending L to any customer $k \neq j$. This property is used in our implementation in order to speed up the dynamic programming algorithm.

4.4.3 Branching rule

As in a branch-and-bound method, the branching scheme in a branch-and-price method consists in extending iteratively the binary search tree by adding constraints implied by the branching rules. In the context of vehicle routing problems, the standard branching rule consists in selecting an arc (i, j) with a fractional flow $0 < f_{ij} < 1$ and then in deriving two branches where in the first branch the use of arc (i, j) is enforced, and it is forbidden in the second. For the multigraph representation, we use a similar branching rule that performs as follows:

- Select an arc $(i, j)^p$ with a fractional flow $0 < f_{(i,j)^p} < 1$ where $f_{(i,j)^p} = \sum_{r_k \in \Omega_t} \alpha^k_{(i,j)^p} z_k$.

  In our implementation, the selected arc $(i, j)^p$ is the first arc such that $0 < f_{(i,j)^p} < 1$, encountered while computing the flow matrix.

- Generate two branches :

  - In the first branch, we enforce the use of arc $(i, j)^p$ in the solution. To do this, we remove from Ω_t all routes using arcs $(i, j)^q$, $q \neq p$, (arcs parallel to $(i, j)^p$), arcs $(i, k)^l$, $k \neq j$, $l = 1, \ldots, |A_{ik}|$ and arcs $(k, j)^l$, $k \neq i$, $l = 1, \ldots, |A_{kj}|$. We remove also all these arcs from the multigraph considered in the pricing phase.
  - In the second branch, we forbid arc $(i, j)^p$. For this aim, we eliminate from Ω_t all routes using arc $(i, j)^p$ and remove this arc from the multigraph considered in the pricing phase.

4.4.4 Stabilization method

As we described in Section 4.4.2, the generated routes and so the search strategy in the column generation method is very dependent on dual values. Frequently, multiple dual solutions are associated with each primal solution. When solving the master problem, the standard function that returns dual values in LP codes returns extreme points of the dual polyhedron. Such solutions provide, most of the times, inappropriate estimations of marginal costs associated with customers, which can lead to very slow convergence [119].

To handle these difficulties, a few techniques are available in the literature that consists mainly in preventing dual variables from taking extreme values and in finding better approximations of the optimal marginal costs (*e.g.*, [119] and [45]). In our algorithm, we implemented the interior point method proposed by Rousseau et al. in [119]. This approach proposes to generate a set of random extreme optimal dual solutions and to provide the subproblem with a convex combination of these solutions.

4.5 Computational experiments

In this section, we present, first, the benchmark problems used in the experiments. Then, we summarize the main results and we analyze the impact of the multigraph representation on solution quality and computing times.

The branch-and-price algorithm is implemented in the C++ programming language. Tests are run on an *Intel CORE i*5 2.6 *GHz* computer with 4GB of memory. We use *CPLEX* 12.6 as the linear programming solver for restricted master problems.

4.5.1 Test data

In our experiments we use four sets of VRPTW instances; the first set of instances is derived from Solomon [126] benchmark instances for the VRPTW, the second set of instances is generated by Letchford et al. in [89] and we used the same procedure proposed in [89] (described below) to generate the third set of instances. The difference between Letchford et al. [89] instances and those we generated is that we used different densities of customers in the road network. In this third set of instances, the number of nodes chosen to be customers is lower, so that the number of possible paths between customers can be higher. The fourth set of instances is generated using real data from the city of *Aix-en-Provence, France*.

4.5.1.1 Modifed Solomon instances

The first set of instances consists of 90 instances derived from Solomon [126] benchmark instances: $R101$ to $R105$, $C101$ to $C105$ and $RC101$ to $RC105$. From each one of these 15 instances, we first generate two graphs, considering only the first 25 and 50 customers, respectively. Then, as in Solomon [126], we set arc costs to the Euclidean distance between customers. Based on these costs and a correlation rule, we propose three different values for travel times: $t_{ij} = \lceil \nu * c_{ij} + \mu * \gamma_{ij} * \check{c} \rceil$ where

- $\check{c} = \max_{(i,j)\in A} c_{ij}$;
- ν and μ are two parameters that define the degree of correlation ($(\nu, \mu) = (0.9, 0.1)$ for a strong correlation between the cost and the travelling distance (SC), $(\nu, \mu) = (0.5, 0.5)$ for a weak correlation (WC) and $(\nu, \mu) = (0.1, 0.9)$ for non-correlated instances (NC)).
- γ_{ij} is a random number in $]0, 1]$.

Other data from the original instances are not modified. For a given original instance and a given number of customers, we thus obtain 3 instances with different levels of correlation.

Note that these correlation rules follow those introduced in [89].

Multigraphs are finally constructed by computing, through dynamic programming, non dominated bi-criterion shortest paths between every vertex pair. Details on the method are given subsequently in subsection 4.5.2.

4.5.1.2 Letchford et al. [89] and Letchford et al.-like (LL) instances

The second set of instances (Letchford et al. [89] instances) were provided to us by the authors. They had been generated with the objective of simulating real-life road networks, with the following procedure: 1. Insert nodes at random positions in the Euclidean space. 2. Consider all possible arcs and insert new arcs sequentially (to represent road segments) on condition that the new inserted arc does not intersect with any other arc and has sufficiently large angles with other arcs at its endpoints. 3. Set arc costs to the Euclidean distance between arc endpoints.

Using this procedure, Letchford et al. [89] generated different sparse graphs with $N \in \{25, 50, 75, 100\}$ nodes. In each graph, one node was selected randomly to be the depot; other nodes were given a probability $p = 0.66$ to be a customer. For each sparse graph, different sets of travel times with different levels of correlation were computed using $t_{ij} = \lceil \nu * c_{ij} + \mu * \gamma_{ij} * \check{c} \rceil$: one non-correlated (NC) travel time matrix, two weakly-correlated (WC), one strongly-correlated (SC) – except for $N = 25$ where one WC travel time matrix was generated instead of two. Then, by mean of a single-source bi-criterion shortest path algorithm, a multigraph was constructed for each case. Note that we did not have to compute these multigraphs because the information was already included in the instance files provided by the authors.

From each of the 15 above multigraphs, 2 instances were derived: a first instance with wide time windows and a second instance with narrow time windows. Time windows were defined such that a set of routes, constructed in a greedy way, were feasible. An integer service time in $\{1, 2\}$ was assigned randomly to each customer. Vehicles were not capacitated. To be consistent with our problem definition, we assigned a demand $d_i = 10$ to each customer and we fixed vehicle capacity to 200. The fleet was considered with a large number of vehicles.

We generated the third set of instances using the same procedure as above, but changing the way customer nodes are defined. Basically, the motivation here was to generate instances with a smaller density of customers. This set consists of 45 instances and is subdivided into three classes of 15 instances with respectively 25 customers / 100 nodes ($p = 0.25$), 50 customers / 100 nodes ($p = 0.5$), 50 customers / 200 nodes. Each class of instances consists of 5 instances where travel times are strongly correlated to costs, 5 instances where travel times

are weakly correlated to costs and 5 instances with no correlation. For each of the obtained road networks, we computed the multigraph through dynamic programming (see 4.5.2). For each multigraph, we completed the VRPTW instance as follows. Each customer was given a service time $s_i = 10$, and a demand $d_i = 10$. Time horizon (time window of the depot $[0, l_0]$) was fixed such that a vehicle can visit any customer and get back to the depot before l_0 using the slowest direct arcs. Time windows were fixed such that each customer can be visited at least on an independent route. The vehicle capacity was fixed to 200 and a fleet with a large number of vehicles was considered.

4.5.1.3 Real instances

Aix-en-Provence is a city-commune in the region of *Provence-Alpes-Cote d'Azur* in the south of France, about 30 km north of *Marseilles*. Based on **OpenStreetMap©** [1] database, we extracted spatial data for two different road networks. The first one (Zone 1, see Figure 4.3) represents the road network of the central urban area. The second (Zone 2, see Figure 4.4) represents the road network in the city center and surroundings.

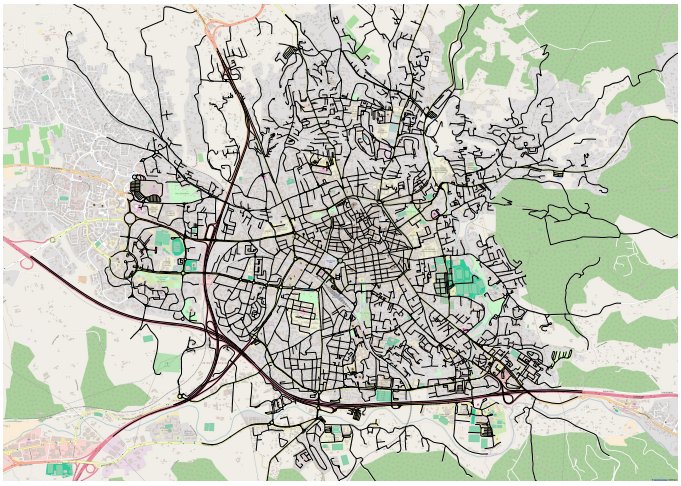


Figure 4.3: *Road Network of the central urban area (Zone 1)*

[1] OpenStreetMap is a collaborative project wich creates and distributes freely available geospatial data. `www.openstreetmap.org/`

Table 4.1: *Real Road networks characteristics*

	Zone 1	Zone 2
Diameter	6 Km	36 Km
# nodes	5437	19500
# arcs	10181	36438

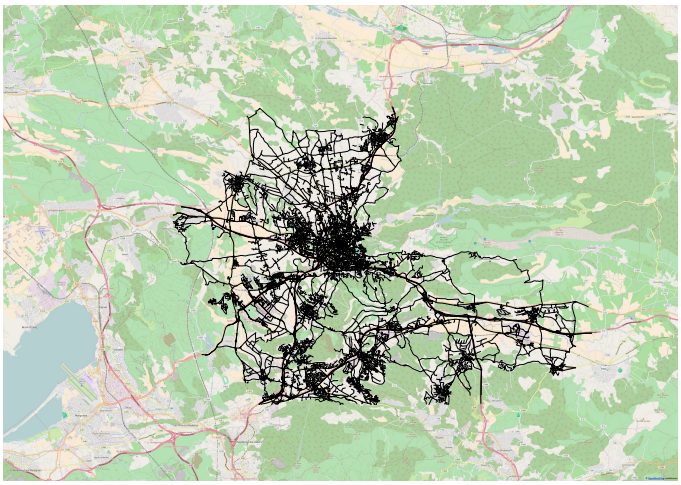


Figure 4.4: *Road Network of Aix-en-Provence center and surroundings (Zone 2)*

Table 4.1 summarizes the characteristics of the extracted road network for the two zones. For each zone, the first row indicates the diameter of the area covered by the extracted road network. The last two rows give the total number of nodes and the total number of arcs.

In each road network, an arc represents a road segment and is defined by a length, a maximum allowed speed and a travel direction. Travel times are computed using road segment speeds and lengths. Costs are set as road segment lengths. A node represents a junction of road segments.

From each of these two road networks, we generated 6 different instances. We considered value set $\{25, 50, 75\}$ for the number of customers, and generated two instances for each case by randomly selecting the customers. For all instances, the depot was selected by hand at a realistic location: near the train station for the first zone and in *Les Milles* industrial area, about 8 Km southwest of *Aix-en-Provence* city-center, for the second zone. Then, we constructed for each instance the associated multigraph as described in subsection 4.5.2. Instance parameters (time windows, service times, demands, vehicle capacity and fleet size) were fixed as for LL instances.

Figures 4.5 and 4.6 represent 2 examples for the selected depot and customers for instances with 25 customers on both road networks.

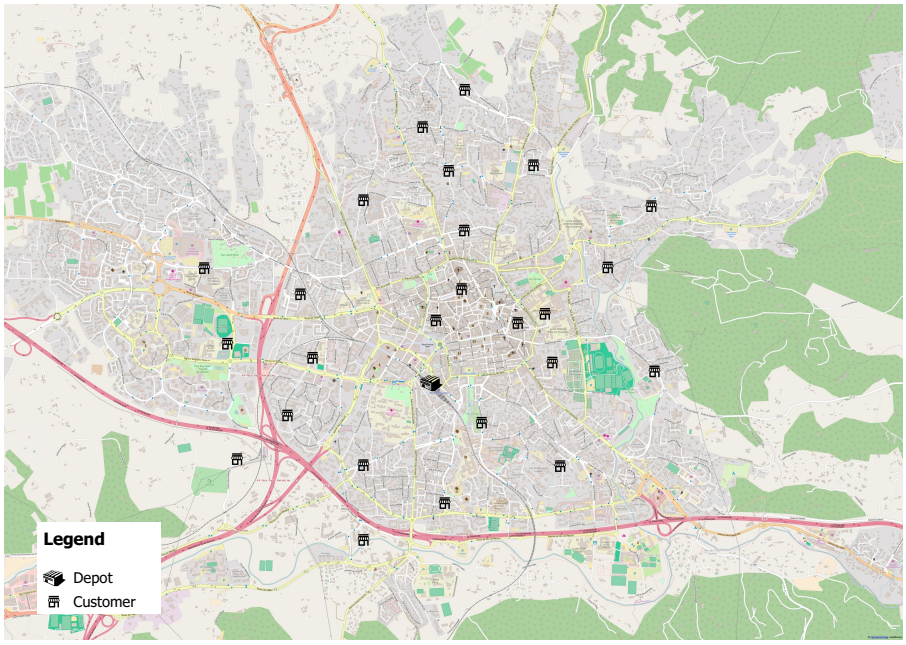


Figure 4.5: *Depot and customers locations for an instance with 25 customers on Zone 1*

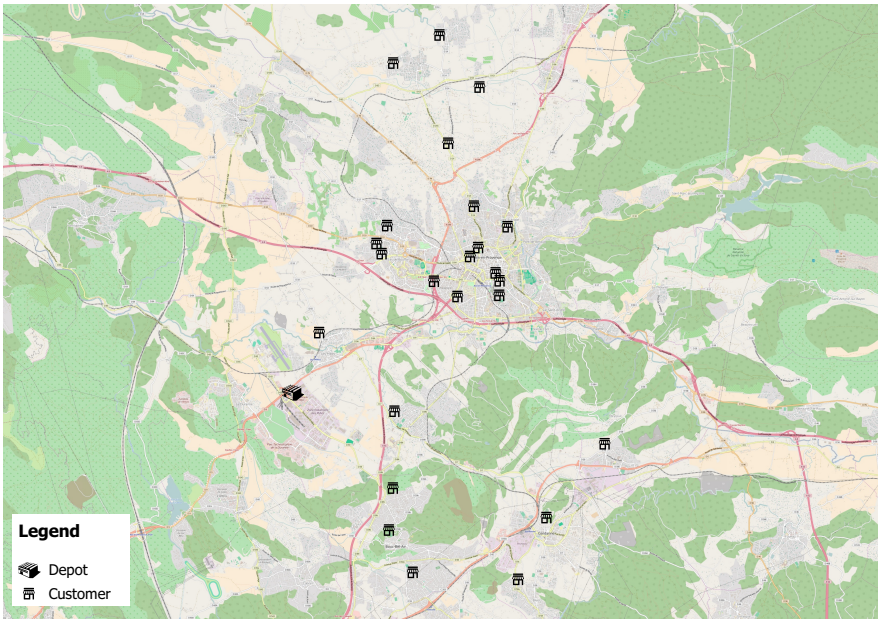


Figure 4.6: *Depot and customers locations for an instance with 25 customers on Zone 2*

4.5.2 Statistics on multigraphs

In this section, we analyze two important pieces of information related to the multigraph construction: computing times and arc set sizes. This information is important to evaluate the tractability of the multigraph representation. Multigraph construction is performed with the preliminary method developed by Ben Ticha et al. [9]. Before starting analyses, we quickly describe the method. For more details, interested readers are referred to [9].

Multigraphs consist of non-dominated paths between all pairs of key-locations. Paths are obtained using a dynamic programming algorithm based on a variant of Dijkstra and A* algorithms. This algorithm aims at solving a bi-objective shortest path problem from a source node to a set of destination nodes. It is repeated $n + 1$ times: one time for each customer and one time for the depot. Basically, the algorithm maintains a set of labels, each one corresponding to a partial path issued from the source node. These labels are sorted according to a "key" that defines the minimum detour of the associated subpath to one of the destinations. Thus, partial path exploration is naturally guided towards destinations, as in the A* algorithm. Bounding mechanisms (based on precomputed mono-objective optimal paths) are introduced to quickly detect bad extensions.

Table 4.2 shows obtained results for modified Solomon instances. The first two columns report the number of customers and the class of original instances. The third column presents the correlation degree. The last three columns report respectively the average number of arcs in the multigraph, the average number of alternative arcs between each pair of locations and the average computing time.

Table 4.3 reports statistics on multigraphs for Letchford et al. [89] instances. The first two columns show the number of nodes and the number of customers. The third column show the correlation degree. The last two columns shows the total number of arcs and the average number of arcs between each pair of nodes for each instance. Computing times are not reported for these instances because we directly received the complete multigraph representation from the authors.

Table 4.4 shows multigraph construction results for LL instances. The first two columns report, respectively, the number of nodes in the original graph and the number of customers. The next column shows the correlation degree. The following columns show, respectively, the average number of arcs in the multigraphs, the average number of alternatives between each pair of customers, and the average time needed to compute the multigraph.

From Tables 4.2 to 4.4, we can observe similar behaviors in modified-Solomon and LL multigraphs. For all configurations (number of nodes in the original graph and number of selected customers), the average number of alternative arcs increases when correlation between travel times and costs decreases. It ranges from 1.6 alternatives for strongly correlated

Table 4.2: *Statistics on multigraph constructions for modified Solomon instances*

# nodes	class	Correlation	# arcs	# alternatives	Time (sec)
25	R	NC	1491	2.5	0.09
		WC	863	1.4	0.07
		SC	613	1.0	0.06
	C	NC	1535	2.6	0.07
		WC	984	1.6	0.07
		SC	636	1.1	0.05
	RC	NC	1742	2.9	0.07
		WC	1185	2.0	0.06
		SC	765	1.3	0.05
50	R	NC	7559	3.1	0.22
		WC	4306	1.8	0.18
		SC	2797	1.1	0.17
	C	NC	7448	3.0	0.20
		WC	4559	1.9	0.18
		SC	2889	1.2	0.17
	RC	NC	8177	3.3	0.19
		WC	5240	2.1	0.18
		SC	3317	1.4	0.18

Table 4.3: *Statistics on multigraph constructions for Letchford et al. [89] instances*

# nodes	# customers	Correlation	# arcs	# alternatives
25	16	NC	372	1.4
		WC	302	1.1
		SC	272	1.0
50	33	NC	1564	1.4
		WC	1624	1.4
		WC	2214	2.0
		SC	1348	1.2
75	50	NC	4452	1.7
		WC	4366	1.7
		WC	6200	2.4
		SC	3100	1.2
100	66	NC	9772	2.2
		WC	8340	1.9
		WC	9844	2.2
		SC	5498	1.2

Table 4.4: *Statistics on multigraph constructions for LL instances*

# nodes	# customers	Correlation	# arcs	# alternatives	Time (sec)
100	25	NC	1682	2.6	0.07
		WC	1231	1.9	0.06
		SC	866	1.3	0.05
100	50	NC	7172	2.8	0.11
		WC	5434	2.1	0.09
		SC	3285	1.3	0.10
200	50	NC	9907	3.9	0.17
		WC	7433	2.9	0.14
		SC	4101	1.6	0.14

Table 4.5: *Statistics on multigraph constructions for real instances*

	# nodes	# customers	# arcs	# alternatives	Time (sec)
Zone 1	5437	25	2109	3.2	4.8
		50	8665	3.4	5.9
		75	22388	3.9	7.1
Zone 2	19500	25	1680	2.7	227.6
		50	7578	3.0	301.7
		75	17000	3.2	367.1

LL instances with 200 nodes and 50 customers to 3.9 alternatives for non-correlated ones. Less alternative arcs exist in Letchford et al. [89] instances. The reason is probably the high density of customers.

Computing times are reasonable and do not exceed 0.25 seconds for all instances. For the second set of instances, one would expect that for the same number of customers, the number of arcs and the computing time increase with the number of nodes in the original road network, which is consistent with the obtained results.

Table 4.5 presents multigraph construction results for real instances. The first two columns specify the considered road network and the number of nodes in the graph. The third column reports the number of customers. The last three columns report the average number of arcs, the average number of alternatives in the constructed multigraphs and the average computing times (in seconds).

A first observation is that the average number of alternatives is in the same order as that observed for the non-correlated version of the previous instances. Also, computing times remain reasonable, even if the size of the road networks are significantly larger: computing times just slightly exceed 6 minutes for the largest instances with 75 customers taken in a 36

Km diameter-wide urban region.

A second observation is that the number of arcs is higher for instances generated from the central urban area's data. A possible reason is that for the first road network, arcs represent short road segments with a high speed variability, thus inducing weakly correlated arc lengths and travel times. Conversely, most arcs in the second road network represent long routes with similar allowed speeds, and so distances and travel times are more correlated than for the first case.

4.5.3 Impact of the multigraph representation

In this section, we investigate the impact of the multigraph representation on solution quality for the VRPTW. To do this, we propose two solution methods:

- The adapted branch-and-price algorithm for the multigraph representation (MG-B&P) described in Section 4.4.
- A branch-and-price algorithm for the customer-based graphs (SG-B&P).

We compare the solution obtained in the multigraph to the solution computed when tackling the problem in a customer-based graph either with cheapest or fastest paths between customers. In the following, we denote by *min-cost graph* the customer-based graph with cheapest paths and by *min-time graph* the customer-based graph with fastest paths. In all experiments, computing times are limited to 7,500 seconds. Obtained results are presented in Tables 4.6 and 4.7 for modified Solomon instances, in Table 4.8 for Letchford et al. [89] instances, in Table 4.9 for LL instances and in Table 4.10 for real instances.

Column "min-cost graph" reports solution costs and computing times in min-cost graphs. Column "min-time graph" reports solution costs and computing times in min-time graphs. Results with the multigraph representation are presented in the third column; column "Gap min-cost" presents the gap between solutions in the multigraph and solutions in the min-cost graph, and column "Gap min-time" presents the gap between solutions in the multigraph and solutions in the min-time graph. The gap is given by

$$Gap(\%) = \frac{cost\ on\ multigraph\ -\ cost\ on\ customer-based\ graph}{cost\ on\ customer-based\ graph} \times 100\% \qquad (4.18)$$

In Table 4.8, column "Instance" presents the instance name which indicates the number of nodes in the original road network, the number of customers in the multigraph and whether time windows are wide (WTW) or narrow (NTW). Column "Corr" reports the correlation level. In Table 4.9 and 4.10, column "Inst" specifies the instance index. Solution costs in Table 4.10 represent total travelled distance and are expressed in meters.

In the following tables, we report only results for relevant instances, i.e., for which the branch-and-price algorithm with the multigraph representation was able to find a feasible solution within 7,500 seconds. In our analyses, we compare the number of obtained solutions using the different graphs for each set of instances.

From Tables 4.6 and 4.7, it comes out that the branch-and-price algorithm solves 67 out of 90 instances with the multigraph representation, while it solves 72 and 74 instances when using min-cost and min-time graphs, respectively. By using a multigraph, costs are reduced up to almost 14% against costs on graphs with the least costly paths (instance $RC101$ with 50 customers and No Correlation, Table 4.7) and costs are reduced up to 54% against costs on graphs with fastest paths (instance $C105$ with 25 customers and No Correlation, Table 4.6). Solution cost is improved for 34 instances when considering only cheapest arcs (48% out of 72 instances solved), the average saving being up to 2.4%, and it is improved for 58 instances compared to solution obtained on min-time graphs (87% out of 74 instances solved), the average saving being up to 14.2%.

We observe that, as expected, increasing the correlation between arc costs and travel times reduces savings provided by the multigraph representation in solution cost. For modified Solomon instances with 25 customers, average "Gap min-cost" and "Gap min-time" range from -4.2% and -35.2% for non-correlated instances to -2.4% and -8.9% for weakly correlated instances and -0.4% and -0.4% for strongly correlated instances (Table 4.6). For modified Solomon instances with 50 customers, average "Gap min-cost" and "Gap min-time" range from -4.7% and -34.6% for non-correlated instances to -3.5% and -10.3% for weakly correlated instances and to -0.2% and -0.8% for strongly correlated instances (Table 4.7).

As expected, computing time increases significantly when using a multigraph. This is due to the fact that for a given node in the branch-and-price search tree the subset of columns to be generated by the pricing problem is larger than when only one arc is considered between each pair of customers. So, the dynamic programming algorithm for the ESPPRC solution is more time consuming with the multigraph.

Table 4.8 summarizes results obtained for Letchford et al. [89] instances. A first observation is that for 8 instances the algorithm cannot find a feasible solution when considering only the least costly paths, while optimal solutions are found on both the multigraph and the min-time graph. This is due to the time windows constraints. A second observation is that, using the multigraph, solution costs are improved for 5 instances compared to solutions on the min-cost graph and for 24 instances compared to solutions on the min-time graph. This improvement can reach 5.4% (instance 100_66_WTW with No Correlation) for the min-cost graph and 11.8% (instance 75_50_NTW with Weak Correlation) for the min-time graph. Obtained solutions on the multigraph are in average 0.8% and 4.1% better than solutions with cheapest paths and with fastest paths. Computing times are more important for the branch-

Table 4.6: *Results for modified Solomon instances with 25 customers*

		min-cost graph		min-time graph		Multigraph		
Corr	Instance	Cost	Time	Cost	Time	Time	Gap min-cost(%)	Gap min-time(%)
NC	R101	690.4	0.1	1281.9	0.1	0.6	0.0	**-46.1**
	R102	594.4	0.3	809.3	0.2	1.7	**-1.0**	**-27.3**
	R103	491.3	1.0	727.7	0.7	12.7	0.0	**-32.5**
	R104	507.3	5.3	676.1	2.4	31.0	0.0	**-25.0**
	R105	653.2	1.0	1002.1	0.2	2.3	**-1.6**	**-35.9**
	C103	223.8	3569.6	364.4	27.0	2197.0	**-10.8**	**-44.6**
	C105	231.9	2.1	485.1	0.3	54.6	**-3.5**	**-53.8**
	RC101	740.2	0.2	1140.1	0.8	0.7	**-10.3**	**-41.1**
	RC102	628.4	0.9	857.4	1.1	10.9	**-12.6**	**-34.9**
	RC103	558	7.3	823.1	2.8	613.9	**-2.2**	**-33.7**
	RC104	441.9	116.2	526.3	84.9	2728.0	**-5.1**	**-20.1**
	RC105	597.3	0.3	798.7	1.2	9.8	**-3.8**	**-27.9**
WC	R101	682	0.2	756.2	0.2	0.2	0.0	**-9.8**
	R102	572.6	0.6	609.8	0.1	1.2	0.0	**-6.1**
	R103	476.2	1.5	504.7	0.3	2.3	0.0	**-5.6**
	R104	481	2.5	499.4	0.6	4.7	0.0	**-3.7**
	R105	601	0.6	669.1	0.4	0.9	0.0	**-10.2**
	C101	262.7	236.7	–	7500	206.5	**-4.6**	**
	C103	199.1	32.9	251.7	34	480.4	0.0	**-20.9**
	C105	216.6	5.4	282.1	7.6	27.3	0.0	**-23.2**
	RC101	609.4	2.4	626.2	1.7	7.7	**-7.9**	**-10.4**
	RC102	627.5	0.6	574.2	38.5	983.5	**-12.0**	**-3.8**
	RC103	473.3	82.4	477.6	19.4	713.3	**-2.4**	**-3.3**
	RC104	399.1	7.5	409.1	22.4	835	**-0.2**	**-2.6**
	RC105	563.7	0.5	602	0.7	2.8	**-1.5**	**-7.7**
SC	R101	684.7	0.1	684.7	0.2	0.2	0.0	0.0
	R102	570.8	0.4	577.2	0.2	0.5	0.0	**-1.1**
	R103	466.6	1.9	458.3	0.3	0.9	**-1.8**	0.0
	R104	420.2	1.9	422.9	0.7	4.2	0.0	**-0.6**
	R105	549.3	0.6	550.1	0.4	0.8	0.0	**-0.1**
	C101	216.6	4.0	220.4	2.6	6.4	0.0	**-1.7**
	C102	193.1	12.4	193.1	0.7	5.5	0.0	0.0
	C103	193.1	45.4	195.2	9.5	96.4	0.0	**-1.1**
	C104	189.7	2299.1	189.7	290	1717.6	0.0	0.0
	C105	194.1	0.5	194.1	0.1	0.5	0.0	0.0
	RC101	532.5	15.7	511.7	4.9	13.9	**-4.7**	**-0.8**
	RC102	443.6	173	443.8	62.3	397.7	0.0	0.0
	RC103	342.2	2.8	342.5	0.5	5.9	0.0	**-0.1**
	RC104	314.9	7.1	314.9	1.4	13.1	0.0	0.0
	RC105	457.6	9.5	458.6	4.6	21.7	0.0	**-0.2**

NOTE *: – indicates that the algorithm has not terminated within 7,500 seconds*

Table 4.7: *Results for modified Solomon instances with 50 customers*

		min-cost graph		min-time graph		Multigraph		
Corr	Instance	Cost	Time	Cost	Time	Time	Gap min-cost(%)	Gap min-time(%)
NC	R101	1322.8	0.7	2274.7	0.4	17.6	**-0.4**	**-42.1**
	R102	1153.8	3.0	1672.3	28.1	52.8	**-0.5**	**-31.3**
	R103	974.7	31.4	1277.6	321	593.7	**-2.3**	**-25.4**
	R104	795	5631.1	–	7500	6798.9	**-3.1**	–
	R105	1169.2	4.8	2041.8	9.3	26.9	**-0.5**	**-43.1**
	RC101	1582.8	17.0	2311.8	3.7	108.0	**-13.1**	**-40.5**
	RC102	1275.1	52	1753.7	10.2	321.6	**-8.7**	**-33.6**
	RC103	1072.7	33.4	1464.2	37.2	6821.9	**-0.9**	**-27.4**
	RC105	1345	7.8	1840	13.4	2358.4	**-8.6**	**-33.2**
WC	R101	1218.8	1.0	1521.8	0.4	1.2	**-3.2**	**-22.5**
	R102	1092.8	3.4	1188.7	1.4	6.3	**-1.6**	**-9.6**
	R103	964.4	74.3	1056.2	20.8	65.4	**-1.7**	**-10.2**
	R104	769.3	893.6	819.7	108.7	1304.3	0.0	**-6.1**
	R105	1066.8	5.3	1162.5	2.8	17.0	**-0.4**	**-8.6**
	RC101	1317.6	5.6	1318.1	16.7	110.3	**-7.2**	**-7.3**
	RC105	1134.1	15.2	1121.7	22.3	68.4	**-8.8**	**-7.8**
SC	R101	1098.3	1.2	1096.3	0.3	1.0	**-1.1**	**-1.0**
	R102	929.8	6.7	935.9	1.2	6.8	0.0	**-0.7**
	R103	827.1	57.1	828.3	19.7	71.1	0.0	**-0.1**
	R105	940.6	7.4	934.8	3.7	14.9	**-0.8**	**-0.2**
	C101	405.4	75.7	407.6	14.9	106.4	0.0	**-0.5**
	C102	366.9	8.4	374.1	9.2	53.8	0.0	**-1.9**
	C103	368.8	770.7	372.7	255.7	699.9	0.0	**-1.0**
	C105	367.9	6.1	369.8	1.7	12.0	0.0	**-0.5**
	RC101	990.9	637.6	1000.8	230.6	2385.9	0.0	**-1.0**
	RC105	940.9	1042.7	947.5	1055.2	6544.4	0.0	**-0.7**

NOTE *: – indicates that the algorithm has not terminated within 7,500 seconds*

Table 4.8: *Results for Letchford et al. [89] instances*

Instance		min-cost graph		min-time graph		Multigraph		
name	Corr	Cost	Time	Cost	Time	Time	Gap min-cost(%)	Gap min-time(%)
25_16_NTW	NC	1252	0.1	1360	0.1	0.1	0.0	**-7.9**
25_16_WTW	NC	1252	0.1	1360	0.1	0.1	0.0	**-7.9**
25_16_NTW	WC	1252	0.1	1265	0.0	0.1	0.0	**-1.0**
25_16_WTW	WC	1252	0.1	1265	0.1	0.1	0.0	**-1.0**
25_16_NTW	SC	1252	0.1	1252	0.1	0.1	0.0	0.0
25_16_WTW	SC	1252	0.1	1252	0.1	0.1	0.0	0.0
50_33_NTW	NC	2184	2.2	2193	0.2	0.5	**-2.2**	**-2.6**
50_33_WTW	NC	2072	91	2094	129.9	398	0.0	**-1.1**
50_33_NTW	WC	2293	0.8	2295	0.6	1.8	0.0	**-0.1**
50_33_NTW	WC	**Infeasible**		2591	0.1	0.6	**	**-5.3**
50_33_WTW	WC	**Infeasible**		2261	7.6	50.6	**	**-4.1**
50_33_NTW	SC	2438	14.5	2445	12.8	19.4	0.0	**-0.3**
50_33_WTW	SC	2104	386.8	2104	260.3	533.9	0.0	0.0
75_50_NTW	NC	**Infeasible**		3653	0.2	0.7	**	**-8.4**
75_50_WTW	NC	3316	84.2	3389	15.7	152.3	**-2.5**	**-4.6**
75_50_NTW	WC	3277	0.9	3315	0.3	1.4	0.0	**-1.1**
75_50_NTW	WC	3262	4.6	3591	0.5	3.2	**-2.9**	**-11.8**
75_50_WTW	WC	3001	161.7	3251	940.3	353.5	**-1.7**	**-9.2**
75_50_NTW	SC	3266	0.9	3270	0.3	1.1	0.0	**-0.1**
75_50_WTW	SC	2949	4666.2	2950	4907.3	5305.9	0.0	0.0
100_66_NTW	NC	**Infeasible**		3642	1.2	64.7	**	**-7.2**
100_66_WTW	NC	3364	1249.6	3500	7500.2	550.1	**-5.4**	**-9.0**
100_66_NTW	WC	**Infeasible**		3569	1.1	6.2	**	**-5.5**
100_66_WTW	WC	**Infeasible**		3449	252.3	4391.6	**	**-6.6**
100_66_NTW	WC	**Infeasible**		3591	1.4	13	**	**-7.9**
100_66_WTW	WC	**Infeasible**		3429	43.1	593.6	**	**-8.0**
100_66_NTW	SC	3319	5	3348	1.0	4.5	0.0	**-0.9**

NOTE *: **Infeasible** indicates that the algorithm can not find a feasible solution with respect to time windows constraints*

Table 4.9: *Results for LL instances*

				min-cost graph		min-time graph		Multigraph		
# nodes	# cust	Corr	Inst	Cost	Time	Cost	Time	Time	Gap min-cost(%)	Gap min-time(%)
100	25	NC	1	1953.3	0.9	1894.3	0.2	6.3	**-6.4**	**-3.5**
			2	2109.6	0.2	2360	0.1	1.4	0.0	**-10.6**
			3	2253.7	0.3	2545.1	0.1	5.7	**-2.3**	**-13.5**
			4	2148.4	0.4	2268.9	0.2	2.8	**-0.4**	**-5.7**
			5	1890.7	0.4	2089.9	0.2	2.5	**-1.1**	**-10.6**
		WC	1	1742.8	0.5	1801.5	0.1	2.0	0.0	**-3.3**
			2	1535.8	11.7	1599	1.2	71.5	**-1.7**	**-5.6**
			3	2109.9	2.6	2065	0.3	3.1	**-2.5**	**-0.4**
			4	1749.7	0.2	1797.7	0.1	0.8	0.0	**-2.7**
			5	2242.8	7.4	2244.1	2.3	10.4	**-3.1**	**-3.2**
		SC	1	2075.4	11.2	2084.2	7.3	21.1	0.0	**-0.4**
			2	2108	0.8	2111.8	0.4	1.3	0.0	**-0.2**
			3	1770.8	3.5	1808.1	1.3	9.7	0.0	**-2.1**
			4	2029.1	0.5	2029.1	0.2	0.6	0.0	0.0
			5	2108.2	0.2	2122.2	0.1	0.6	0.0	**-0.7**
	50	NC	1	2613.6	233.5	2810.2	21.7	242	**-1.9**	**-8.8**
			2	3483	1825.2	3404.7	50.4	2296.5	**-4.7**	**-2.5**
			3	2843.5	179.6	2820	33.5	1045.6	**-4.0**	**-3.2**
			4	2725.4	92.2	2812.4	63.4	399.9	**-4.0**	**-7.0**
			5	3031.3	79.3	3156.5	5.6	82.2	**-2.7**	**-6.6**
		WC	1	2626.8	16.7	2776.3	5.0	104.8	0.0	**-5.4**
			2	2981.9	25.2	2983.4	15.2	245.5	**-3.1**	**-3.1**
			3	2598.2	25.8	2563.7	13.3	79.1	**-3.1**	**-1.8**
			4	2398.3	18.3	2516.8	8.9	42.2	0.0	**-4.7**
			5	2427.2	531	2476.2	5.0	291.3	0.0	**-2.0**
		SC	1	3188.1	104.8	3177.9	53.5	1120.1	**-0.3**	0.0
			2	3116.5	205.1	3118.1	62.5	424.2	0.0	**-0.1**
			3	3305.7	21.3	3177.3	2.4	21.4	**-4.0**	**-0.1**
			4	2977.5	23.6	2993.1	6.5	44.9	0.0	**-0.5**
			5	3352.2	4.1	3365.4	0.9	5.4	0.0	**-0.4**
200	50	NC	1	4469	84	4396.2	5.6	1659.3	**-7.7**	**-6.2**
			2	4200.9	105.8	4511.3	5.9	191.3	**-4.8**	**-11.3**
			3	4458.2	64.8	4619.3	14.4	1045.1	**-4.0**	**-7.4**
			4	4309.6	719.6	4341.9	128.9	6775.5	**-5.6**	**-6.3**
		WC	1	4514.3	2544.4	4644	66.2	4378.7	**-3.5**	**-6.1**
			2	4035.2	13.9	4140.3	18.9	181.7	**-3.5**	**-5.9**
			3	4221.7	52.3	4270.7	29.6	651.6	**-4.1**	**-5.2**
			4	3833.2	3501.6	3936.7	24.3	695.4	**-3.9**	**-6.4**
			5	4746.5	1944.6	4718.9	33.8	453.4	**-8.8**	**-8.3**
		SC	1	4416.8	937.6	4461	423	3892.8	0.0	**-1.0**
			3	4282.3	163.8	4295.7	46.4	203.3	0.0	**-0.3**
			4	3719.8	92.9	3741.1	25.0	81.8	0.0	**-0.6**
			5	3765.7	59.9	3773.7	13.9	67.5	0.0	**-0.2**

Table 4.10: *Results for Real instances*

			min-cost graph		min-time graph		multigraph		
	# cust	Inst	Cost	Time	Cost	Time	Time	Gap min-cost(%)	Gap min-time(%)
Zone 1	25	1	46535	0.2	48705	0.3	1.7	**-3.4**	**-7.7**
		2	48425	0.2	47570	0.2	0.8	**-8.0**	**-6.3**
	50	1	81816	1.9	84320	4.3	13.4	**-2.3**	**-5.2**
		2	86119	2.9	88616	3.7	18.8	**-1.6**	**-4.4**
	75	1	111283	23.4	116668	21.2	131.4	**-0.5**	**-5.1**
		2	102448	311.6	108065	11.4	73.4	**-0.7**	**-5.9**
Zone 2	25	1	132357	1.5	137335	0.2	1.2	**-6.6**	**-10**
		2	195992	0.2	211069	0.4	1.1	**-1.7**	**-8.7**
	50	1	272062	4.0	300284	12.6	22.7	**-0.1**	**-9.5**
		2	370837	10.4	395135	13.6	13.3	**-2.3**	**-8.3**
	75	1	436378	599.1	407309	1372.3	174.1	**-10.5**	**-4.1**
		2	378277.0	55.4	392087.0	23.6	102.6	**-0.9**	**-4.4**

Table 4.11: *Average gaps (%) obtained with LL instances*

		min-cost graph			min-time graph		
# nodes	# customers	NC	WC	SC	NC	WC	SC
100	25	-2.1	-1.5	0.0	-8.8	-3.0	-0.7
100	50	-3.5	-1.2	-0.9	-5.6	-3.4	-0.2
200	50	-5.5	-4.7	0.0	-7.8	-6.4	-0.5

and-price algorithm with a multigraph representation than with both customer-based graphs. The average computing time for the solved instances on the multigraph is 461 seconds while it is 351 and 366 seconds for the same set of instances on the min-cost graph and the min-time graph respectively. Another important observation is that gaps tend to be smaller when time windows are wide.

For LL instances, Table 4.9, we see that by using a multigraph, 25 solutions on the min-cost graph and 42 solutions on the min-time graph are improved. Costs are reduced by up to 8.8% and 13.5% compared to solutions obtained on the min-cost graph and the min-time graph respectively. The average savings are 1.2% and 4.2% for instances with 25 customers and 100 nodes, 1.9% and 3.1% for instances with 50 customers and 100 nodes and 3.5% and 5% for instances with 50 customers and 200 nodes. Clearly, the saving in solution cost increases when the density of customers in the road network decreases. This can be explained by the fact that by decreasing the density of customers, for the same number of nodes in the road network, the number of alternative arcs between each pair of nodes in the multigraph increases (see Table 4.4, thus the solution space is larger and more potentially good solution may be obtained.

Table 4.11 shows the impact of the correlation magnitude on the reduction of solution costs for LL instances. Clearly, for the same numbers of nodes and customers, decreasing

the correlation between arc costs and travel times increases the reduction in solution costs for both customer-based graphs. One also notices that for all classes of instances, cost savings are higher for solutions on min-time graphs than for solutions obtained on min-cost graphs.

Computing times are higher with the multigraph representation; the branch-and-price algorithm could not find feasible solutions for 2 out of 45 instances, while by using both customer-based graphs optimal solutions are always found. The average computing times with the multigraph, min-cost graph and min-time graph are 606.7, 309.8 and 27.1 seconds, respectively.

In Table 4.10, we summarize results obtained for real instances. We notice that, by using a multigraph representation, we obtained negative gaps for all instances compared to both customer-based graph models. Solution costs are reduced by up to 10.5% compared to solutions on min-cost graphs (instance 1 with 75 customers in the second road network (Zone 2)), and are reduced by up to 10% compared to solutions on min-time graphs (instance 1 with 25 customers in the second road network (Zone 2)). By using the multigraph, better solutions are obtained; the average saving in solution cost is about 3.2% for min-cost graphs and 6.6% for min-time graphs. As observed for other sets of instances, computing times are, generally, more important with the multigraph model.

Overall, these results illustrate the important impact of the multigraph representation on VRPTW solution quality. By using a multigraph, significant savings are obtained both with randomly generated and real instances.

4.6 Conclusion

In this paper, we were interested in the Vehicle Routing Problem with Time Windows on road networks, where several attributes are defined for each route segment. We proposed to model the problem using a multigraph representation where we considered between each pair of customers all possible paths. To investigate the interest of our approach, we developed a branch-and-price Algorithm for the multigraph based VRPTW. The computational experiments showed the positive impact of the multigraph representation and attractive savings in solution values for a large number of benchmark instances are attained. It also demonstrated the tractability of the approach.

On a more general level, our primary interest was to emphasize the limits of the standard model used to tackle a large class of vehicle routing problems. The VRPTW provided a typical example, but many other routing problems also involve several attributes on road network segments. Our results confirm and complete results already reported in Garaix et al. [61] and in Lai et al. [85] for other categories of problems.

As a perspective of this work, one may be interested in developing an efficient heuristic solution method adapted to the multigraph representation. The results obtained here showed that the multigraph representation increases significantly computing times. Hence, a relevant issue would be to follow Lai et al. [85] and develop dedicated solution techniques for the multigraph.

A second perspective would be to investigate the solution of these categories of problems directly on the original road network without constructing the multigraph. As we mentioned, a first paper in this regard was proposed by Letchford et al. [89]. In this paper, a numerical comparison between the two approaches (multigraph approach and road network approach) is presented. According to Letchford et al. [89], tackling the problem directly with the road network is more efficient, at least at the root node of the branch-and-price tree (the complete branch-and-price scheme was not developed). We started some preliminary experiments and obtained very contrasted results that did not necessarily confirm the conclusions drawn in Letchford et al. [89]. The reason might be that Letchford et al. [89] consider instances with a high density of customers and allow non-elementary routes in the linear programming relaxation. Further analyses and extensive comparisons are thus important to achieve a clear understanding of the relative efficiency of the two approaches.

Chapter 5

Adaptive Large Neighborhood Search for the Vehicle Routing Problem with Time Windows with a multigraph representation for the road network

This chapter is submitted for a publication in *Computers & Operations Research* journal.

Abstract

In this paper, we study the Vehicle Routing Problem with Time Windows (VRPTW) on road networks where several attributes are defined on road segments. This problem is generally addressed using a complete graph representation, so-called customer-based graph. An arc represents the shortest origin-destination path computed using a single criterion. Such a representation could be disadvantageous because some alternative routes that propose different compromises could be discarded from the solution space. Instead, we use a special representation of the road network called a multigraph. In this representation, an arc is introduced for every non-dominated path connecting the same origin-destination nodes. We develop an Adaptive Large Neighborhood Search (ALNS) heuristic in which a special data structure and a dynamic programming algorithm are used to efficiently address the multigraph setting. Computational experiments on several set of instances demonstrate the effectiveness of our solution method and the impact of alternative routes on the solution quality.

Keywords: *Routing, Multigraph, Heuristics, Dynamic programming, Road network.*

5.1 Introduction

Distribution activity presents one of the most important components in logistic systems. It is estimated that distribution costs represent almost half of the total logistic costs and that for

some industries it can account up to 70% of the total cost of goods [33]. Therefore, improving vehicle routing strategies leads to important logistic cost savings.

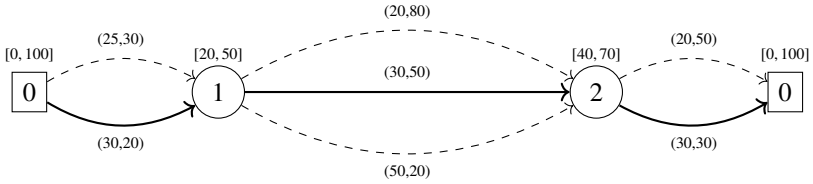
The Vehicle Routing Problem (VRP) was introduced first by Dantzig and Ramser [31] in 1959. Basically, the VRP aims at computing a set of minimum-cost vehicle routes that start and end at a depot. Each customer is supplied exactly once and each route satisfies a set of constraints: vehicle capacity, time windows, route duration, etc.

Most of proposed approaches are based on a complete graph representation for the underlying road network, so-called customer-based graph. A node is introduced for each point of interest and arcs represent shortest paths linking these points. These shortest paths are, typically, computed according to a single criterion (travel cost, travel time, distance, etc.).

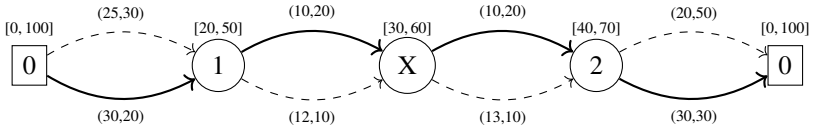
In many cases, road segments in the original network are defined with several attributes. Using a customer-based graph in such situations could discard, from the solution space, some alternative routes with different trade-offs [61, 8]. Few works in the literature evoked this issue and proposed solution approaches to handle it. One of these approaches consists in representing the road network with a multigraph: an arc is introduced for each possible origin-destination path in the original network. This approach was investigated by Garaix et al. [61] in the context of an On-Demand Transportation Problem. They showed the interesting impact of the multigraph representation on the solution quality. Ben Ticha et al. [8] generalized these results, they confirmed that the classical complete graph representation can have a negative effect on the solution quality for vehicle routing problems with several attributes on road segments.

In this paper, we study the Vehicle Routing Problem with Time Windows (VRPTW) based on a multigraph representation. The VRPTW aims at planning a set of vehicle routes that satisfy customer requests within their time windows. We study the problem on a multigraph where parallel arcs between two nodes represent non-dominated paths connecting the two customer locations. These paths are computed according to travel costs and travel times. We propose a heuristic method based on an Adaptive Large Neighborhood Search (ALNS) to solve the problem. Such a heuristic has shown its efficiency for a large number of vehicle routing problems. The main idea in a Large Neighborhood Search (LNS) is to reschedule some customers service in a solution at each iteration in order to evaluate many different solutions. Thus, different areas from the solution space could be explored. However, as shown by Garaix et al. [61], even simple operations such as removals and insertions of customers become difficult to evaluate with a multigraph setting, since the sequence of arcs that links customers along a route must be re-optimized at each iteration. To illustrate this issue, let us consider the example presented in **Figure 5.1**.

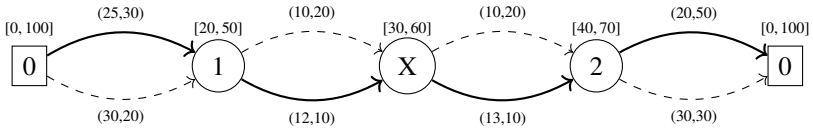
In **Figure 5.1a**, we are given an initial route defined by the sequence of nodes $\{0, 1, 2, 0\}$. Each node is associated with a time window. Parallel arcs between each pair of nodes are



(a) *Initial route*



(b) *Route after a customer insertion without re-optimizing the selected arcs*



(c) *Optimal route after a customer insertion*

Figure 5.1: *Illustration of a customer insertion*

provided, each arc is defined by a cost and a travel time given between parentheses, in this order *(cost, time)*. Selected arcs in the route are represented with a thick line. Suppose that we want to evaluate the insertion of customer X between customers 1 and 2. Typically, this is done by selecting the best arcs linking 1 with X and X with 2 and that ensure the feasibility of the new route regarding time window constraints. In this case, the obtained route is provided in **Figure 5.1b** and has a total cost equals to 80. However, if we recompute the whole sequence of arcs to be used to link the new sequence of nodes $\{0, 1, X, 2, 0\}$, a new route (see **Figure 5.1c**) is obtained with a better total cost equals to 60.

Garaix et al. [61] proved that for a given sequence of nodes, computing the optimal sequence of arcs is an NP-hard Problem. We develop a new procedure that provides the optimal routes for new sequences of customers and thus, allows to efficiently evaluate neighbour solutions.

The main contribution of this paper is that it proposes an efficient heuristic method for the multigraph-based VRPTW. The VRPTW is one of the most studied variant of vehicle routing problems due to its various applications in practice. Considering alternative paths permits to address more efficiently real-life applications where some compromises between the minimum cost and the good service quality, for example, would be interesting for a distributor. However, such a modelling could induce some computational challenges, especially in exploring the neighbourhood of a given solution. Applying solution methods with standard settings could lead to an overestimation of solution costs (as illustrated in **Figure 5.1**).

We propose a procedure based on an incremental data structure and a dynamic programming algorithm to efficiently handle the multigraph setting. Another contribution of this paper is that it presents an empirical analyses on the impact of the multigraph representation on the solution quality. We conduct a computational study based on several types of benchmark problems (modified instances from the literature, realistic instances and instances derived from real data). Obtained results show that using the proposed solution method considerable savings are obtained in reasonable computing times.

The reminder of this paper is organized as follows. In Section 5.2 we review the related literature. In Section 5.3, a mathematical formulation for the multigraph based VRPTW is presented. A solution method is described in Section 5.4, with a focus on main features proposed to handle the multigraph setting. Finally, computational analyses are conducted in Section 5.5 to evaluate the performance of the proposed heuristic and the impact of the multigraph representation on the solution quality.

5.2 Literature review

Since its introduction in 1959 by Dantzig and Ramser [31], the Vehicle Routing Problem was the subject of intensive research efforts. Many books and articles reviewed this literature and provided the main properties and proposed solution methods for the classical VRP (e.g. [65], [128], [87] and [132]).

Due to the large number of applications where routing issues are involved, many VRP variants with different characteristics and additional constraints were studied in the literature. One of the most studied VRP variant is the well-known VRPTW. This variant differs from the classical VRP in that each customer has to be served in a specified time window. The VRPTW has drawn many researchers' attention and a large number of solution methods are proposed in the literature. Baldacci et al. [2] and Cordeau et al. [25] reviewed the literature related to exact solution algorithms. Kallehauge [78] focused on mathematical formulations in relation with exact methods and gave an analysis of the polyhedral structure of the VRPTW. Construction heuristics and local-improvement methods were reviewed in [14] and main meta-heuristics developed in the literature were discussed in [15].

In this work, we consider an additional feature where parallel arcs between each pair of customers are considered: an arc is introduced for every non dominated path in the underlying road network. These arcs are computed based on the different attributes of road segments and could provide more flexibility in route construction (for example, a carrier may prefer faster but more costly connections when delivery times are restricted).

Although original, the use of parallel arcs in the context of vehicle routing problems

is not entirely new. Baldacci et al. [3] used the multigraph structure to tackle a Multiple Disposal Facilities and Multiple Inventory Locations Rollon-Rolloff Vehicle Routing problem (M-RRVRP). The authors showed that this problem is equivalent to a constrained time vehicle problem on a multigraph. They proposed an iterative exact method based on a set partitioning formulation. The considered multigraph in [3] is a representation of the original graph associated with the M-RRVRP where nodes represent feasible trips and an arc (i, j) is introduced for every possible sequence of moves that a vehicle can do in order to perform trip j immediately after trip i.

In a different context, Caramia and Guerriero [17] used a multigraph representation to tackle a Long-Haul Freight Transportation Problem stimulated by a real-life application and where routes may be multimodal. Their objective was to determine a transportation plan that minimizes the total travel time and route cost and maximizes a transportation-mean sharing index. Based on the network definition, they observed that alternative routes could exist between two points and that the transportation network could be seen as a multimodal multigraph; some arcs with the same endpoints can be traversed using the same transportation mode but have different costs and travel times and some arcs have the same endpoints but are associated with different transportation modes. They showed that such structure enlarges significantly the solution space and makes the definition of the best solution for the multi-objective scenario very difficult.

A more relevant study was proposed by Garaix et al. [61]. Motivated by the development of a real-world on-demand transportation system, they sought to efficiently solve a Dial-a-Ride Problem. They pointed out that when several attributes are defined on road segments, transforming a VRP on a road network to a standard VRP could be disadvantageous and may discard potentially good routes from the solution space. They introduced a multigraph representation where parallel arcs represent the set of Pareto-optimal shortest paths and proposed two solution methods: a simple insertion heuristic and an exact method based on a branch-and-price procedure. The authors showed that when using multigraph representation, some difficulties arise. In particular, they showed that defining the optimal route for a given sequence of customers is NP-hard problem and called it Fixed Sequence Arc Selection Problem. To solve this problem, they developed a dynamic programming algorithm and embedded it in their heuristic.

A new generalization of the VRPTW was introduced by Reinhardt et al. in [113] in which additional fixed costs are associated with subsets of edges in the road network. Their study was motivated by the fact that in some real-life situations, the access to some roads, areas or bridges implies additional fees. These fees may correspond to toll roads, ferry connections or special certifications for access to war zones or areas of unrest. In such situations, connection costs between customer locations do not depend only on traveling costs, but also on buying or not access to a subset of connections in the transportation network. The authors called this problem Edge Set Vehicle Routing Problem with Time Windows where edges are regrouped

in different subsets and a fixed additional cost is associated with each subset. Reinhardt et al. [113] stated that this problem can be modelled using a multigraph representation where parallel arcs represent edges connecting the same customers but belonging to different edge subsets. They proposed a Branch-Price-and-Cut method to solve a simplified version of the problem where only one edge exists between each pair of nodes. The multigraph case was left as a perspective.

Lai et al. [85] investigated the impact of a multigraph representation on the Time-Constrained Heterogeneous Vehicle Routing Problem (HVRP). They proposed a tabu search heuristic that can handle the neighbourhood exploration with multigraph structure. Experiments were carried out on randomly generated instances with two alternative arcs between each pair of nodes.

Later, Ben Ticha et al. [8] studied the impact of the multigraph representation on the solution quality for vehicle routing problems when several attributes are defined on road network arcs. They considered the VRPTW as a test-bed problem. They proposed an exact method based on a Branch-and-Price procedure. An experimental analysis was conducted on several types of instances (benchmark instances from literature and instances derived from real road networks data). They demonstrated the interesting impact of the multigraph representation on the quality of solution compared to the complete graph representation.

This literature review confirms that only a few papers investigated vehicle routing problems on multigraphs with relatively different goals. Garaix et al. [61], Lai et al. [85] and Ben Ticha et al. [8] confirmed that the customer-based graph can lead to an overestimation of the solution cost for vehicle routing problems when several attributes are defined on road segments. They showed that the multigraph representation has an interesting impact on the solution quality.

In this study, we propose to develop an efficient solution method and to conduct further analyses. Note that Garaix et al. [61] and Ben Ticha et al. [8] based their conclusions on exact solutions. The VRPTW is one of the most difficult optimization problems and the multigraph structure could increase significantly its complexity. Consequently, exact solution methods may fail to solve realistic-sized problems. Therefore, we propose a heuristic method that can address efficiently the multigraph structure. Lai et al. [85] based their comparison on results obtained with a tabu search heuristic and randomly generated instances with only two alternative arcs between each two nodes. In this work, we propose to conduct an intensive experimental study based on different sets of instances: modified benchmarks from the literature and instances derived from real road networks.

As we mentioned before, when using the multigraph representation, applying an ALNS scheme with standard settings could be misleading. An important issue is how to evaluate neighbors solutions: each solution consists of new sequences of customers for which best

arc selections have to be computed. This task involves an NP-hard problem. Garaix et al. [61] and Lai et al. [85] proposed, respectively, an exact and heuristic algorithms to define arc selection decisions. Using these algorithms, the whole sequence of arcs is recomputed each time a new sequence of nodes has to be evaluated. This may be very time consuming. In order to avoid this issue, we develop a new procedure in which only local decisions related to arc selection have to be evaluated. This procedure (described in section 5.4.1) is based on an incremental data structure and a dynamic programming algorithm.

5.3 Problem Formulation

The multigraph-based VRPTW is defined on a directed multigraph $G = (V, A)$ consisting of V a set of $n + 2$ nodes and A a set of arcs. G derives from a road network as follows: nodes 0 and $n + 1$ represent the depot location and a node i is introduced for each customer location $1 \leq i \leq n$. The multi-arc set A contains parallel arcs between every pair of nodes: $A = \bigcup_{(i,j)\in V^2} A_{(i,j)}$ where $A_{(i,j)} = \{(i, j)^p, p = 1, \ldots, m_{ij}\}$ represents the set of m_{ij} alternative paths connecting two customer locations i and j in the underlying road network. All feasible routes correspond to paths in G starting at node 0 and ending at node $n + 1$.

We associate with each customer node a demand d_i, a time window $[e_i, l_i]$ and a service time st_i (with $d_0 = d_{n+1} = 0$, $e_0 = e_{n+1}$, $l_0 = l_{n+1}$, and $st_0 = st_{n+1} = 0$). We associate with each arc $(i, j)^p$ a travel cost $c_{(i,j)^p}$ and a travel time $t_{(i,j)^p}$ that represent, respectively, the cost and the time needed to go from customer location i to customer location j through the associated path indexed by p in the road network. We consider a homogeneous fleet $\mathcal{K}$ with K vehicles and a loading capacity Q. Our objective is to compute a set of feasible routes (that satisfy customer time windows, vehicles capacity) that serves each customer exactly once and that minimizes the total travelling cost.

We define the following two decision variables: $x^k_{(i,j)^p}$: is a binary variable equals to 1 if vehicle k travels on arc $(i, j)^p$ and equals to 0 otherwise, and t^k_i: denotes the starting time of service at customer i if it is served by vehicle k. The multigraph-based VRPTW can be formulated as follows:

$$Min \sum_{k\in K} \sum_{(i,j)^p \in E} c_{(i,j)^p} x^k_{(i,j)^p} \tag{5.1}$$

$$s.t. \quad \sum_{k\in K} \sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} x^k_{(i,j)^p} = 1 \qquad i \in C \tag{5.2}$$

$$\sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} x^k_{(j,i)^p} - \sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} x^k_{(i,j)^p} = 0 \qquad i \in V, k \in K \tag{5.3}$$

$$\sum_{i\in V} \sum_{p=1}^{|A_{(0,i)}|} x^k_{(0,i)^p} = 1 \qquad k \in K \tag{5.4}$$

$$\sum_{i\in V} \sum_{p=1}^{|A_{(0,i)}|} x^k_{(i,n+1)^p} = 1 \qquad k \in K \tag{5.5}$$

$$\sum_{i\in V} \sum_{j\in V} \sum_{p=1}^{|A_{(i,j)}|} d_i x^k_{(i,j)^p} \leq Q \qquad k \in K \tag{5.6}$$

$$t^k_i + s_i + t_{(i,j)^p} x^k_{(i,j)^p} \leq t^k_j + M(1 - x^k_{(i,j)^p}) \qquad i, j \in V, 1 \leq p \leq |A_{(i,j)}, k \in K \tag{5.7}$$

$$e_i \leq t^k_i \leq l_i \qquad i \in V, k \in K \tag{5.8}$$

$$x^k_{(i,j)^p} \in \{0, 1\} \qquad i, j \in V, 1 \leq p \leq |A_{(i,j)}|, k \in K \tag{5.9}$$

$$t^k_i \geq 0 \qquad i \in V, k \in K \tag{5.10}$$

The objective in (5.1) is to minimize the total traveling cost. Constraints (5.2) ensure that each customer is visited by a single vehicle exactly once. Constraints (5.3) ensure that each vehicle arriving at a customer leaves to another and constraints (5.4) and (5.5) ensure that each vehicle starts at the depot (node 0) and finishes at the depot (node $n + 1$). Inequalities in (5.6) guarantee that the total demand served through the vehicle route k do not exceed its capacity. Constraints (5.7) establish the service timetable at customers in the same route and permit to eliminate sub-tours. Finally, constraints (5.8) guarantee the respect of time windows. Constraints (5.9) and (5.10) define decision variables.

5.4 Solution Method

The VRPTW has been intensively studied and many heuristic optimization approaches have been developed in the literature [14]. Most of these heuristics are based on local search procedures that start from an initial solution and improve it iteratively by exploring, at each step, close solutions. To evaluate neighbouring solutions, small changes are made to the current one, by moving one or more customers, exchanging two or more services between two routes,

etc. A huge number of solutions can be explored in a short time. Given that the VRPTW is a highly constrained optimization problem, a local search approach may have difficulties in moving from one promising area of the solution space to another. Obtained solutions are generally local optima and have a tight structure. To avoid getting stuck in local optima, we opt for the use of move-generation mechanisms that can search a larger neighbourhood, so that we can explore different promising areas from the solution space.

Our heuristic is based on an Adaptive Large Neighborhood Search (ALNS) procedure, introduced by Ropke and Pisinger [118] to solve the Pickup and Delivery Problem with Time Windows (PDPTW). The ALNS heuristic is itself a variant of the Large Neighborhood Search heuristic (LNS) proposed by Shaw [124] to solve the VRPTW. The main idea of the LNS heuristic is to rearrange, at each iteration, a large part of a solution using a combination of a *destroy* and a *repair functions*. The destroy function consists in removing a predefined number of customers. The repair function aims to reinsert the removed customers in order to create a new feasible solution. In a local-search scheme, the algorithm examines the neighbourhood of the best solution found at each iteration. However, the LNS may search in the neighbourhood of non-improving solutions when such solutions are accepted in the search procedure. The goal of this strategy is to avoid premature convergence toward local optima and to force search procedure to visit different areas from the solution space. While the LNS heuristic uses only one destroy procedure and one repair procedure, the ALNS uses several removal and insertion heuristics and applies them alternatively using a selection mechanism that considers the statistics obtained during the search, hence the use of the term *Adaptive*. The general scheme of the ALNS heuristic is described by **Algorithm 5.1**.

Algorithm 5.1 Adaptive Large Neighborhood Search

```
1:  Compute an initial solution solution s_curr
2:  Initialize s_best ← s_curr
3:  while stopping criteria not met do
4:      Select a destroy procedure d ∈ D and a repair procedure r ∈ R
5:      s ← r(d(s_curr))
6:      if accept(s, s_curr) then
7:          s_curr ← s
8:          if c(s) < c(s_best) then
9:              s_best ← s
10:         end if
11:     end if
12: end while
13: return s_best
```

Line 5 in **Algorithm 5.1** specifies the move generation function i.e. the function used to search the neighborhood of a solution s_{curr}. This function consists of a sequence of elementary operations: removal operations in the destroy phase and insertion operations in

the repair phase. Elementary operations involves different types of decisions: assignment and sequencing decisions i.e. allocating customers and defining service order for each vehicle (essentially for insertion operations) and scheduling decisions i.e. computing the visit timetable for customers in the same route. When tackling the standard VRPTW, scheduling decisions are trivially deduced once other decisions are fixed. This property doesn't hold using a multigraph representation. Additional decisions related to the selection of arcs to be used along each route have to be taken. Garaix et al. [61] showed that for a given sequence of nodes, computing the optimal sequence of arcs is an NP-hard Problem so-called Fixed Sequence Arc Selection Problem (FSASP). In order to correctly define scheduling decisions for the mutligraph case, we propose a new procedure based on an incremental data structure and a dynamic programming algorithm. This procedure provides the optimal selection of arcs to be used to link a new sequence of customers obtained during the search scheme.

In the remainder of this section, we first introduce the data structure and the algorithm used to handle arc selection decisions in Section 5.4.1. In Section 5.4.2 we present an *Adapted Savings Algorithm* that we use to construct an initial solution. Removal and insertion heuristics are then described in Sections 5.4.3 and 5.4.4. In Section 5.4.5, we present a selection mechanism that we use to determine the removal and insertion heuristics to apply at each iteration. Finally, we present the acceptance criterion in Section 5.4.6.

5.4.1 Arc selection procedure

In neighborhood search heuristics for VRP, elementary operations are frequently performed in the search procedure. Here, a number of customers are removed then reinserted at each iteration. As mentioned before, these operations, in presence of parallel arcs, involve an NP-hard problem. For every new sequence of nodes, we have to solve a FSASP i.e., we have to determine the sequence of arcs to link each pair of consecutive nodes. To deal with this problem, Garaix et al. [61] proposed a dynamic programming algorithm based on the approach introduced by Irnich and Desauchiers [74] (to solve the *Shortest Path Problem with Resources Constraints*). Later, a heuristic method was proposed by Lai et al. [85] based on a local search procedure.

Note that, with both methods cited above, one has to recompute the set of arcs to be used for every new sequence of nodes. In a LNS scheme, a large number of elementary moves have to be evaluated at each iteration and an arc selection procedure has to be applied for each evaluation. This may be time consuming even using a heuristic procedure. In this section, we propose a procedure based on an incremental data structure to handle this issue, so that we do not have to recalculate the whole sequence of arcs when evaluating or performing an elementary operation such as removal or insertion of a customer.

Our data structure is based on a bidirectional labeling approach. Each node $v \in V$ is

associated with a set of forward and backward labels. Given a sequence of nodes S to be visited in the same route, a forward label associated with a node $v \in S$ represents a partial route from the depot location to v and a backward label represents a partial route from v to the depot location. A label is defined with the following information $L = (c, t, q, v)$ with:

- c: travel cost of the partial route represented by the label;
- v: last (resp. first) node visited along the partial route represented by the forward (resp. backward) label;
- t: earliest (resp. latest) starting service time at node v in the partial route represented by the forward (resp. backward) label;
- q: total demand to be delivered along the partial route represented by the label.

In the following, we denote by $S = (v_0, v_1, ..., v_{p+1})$ a sequence of nodes visited along the same route where v_{p+1} and v_0 correspond to depot location. $G_S = (V_S, A_S)$ denotes the linear multigraph obtained from G by considering in A_S only arcs connecting each pair of consecutive nodes in S and $V_S = \{v_0, ..., v_{p+1}\}$. Let $R(S)$ denotes the optimal route induced by the sequence S i.e., $R(S)$ corresponds to the least cost selection of arcs from A_S such that each customer is visited within his time window along $R(S)$.

Given an initial solution, our procedure starts by computing the set of forward and backward labels arriving at each node in V using **Algorithms 5.2** and **5.3**. An *Adapted Savings algorithm* (see Section 5.4.2) is used to construct an initial solution.

Forward and backward labels are stored and maintained in vectors (resp., FL and BL) during the entire search procedure. For each node i, $FL[i]$ and $BL[i]$ are two lists of labels sorted in a non-decreasing order of the travel cost c. We denote by $FL[i]_k$ (respectively $BL[i]_k$) the k^{th} label in $FL[i]$ (in $BL[i]$) according to this order. Label $FL[i]_0$ and $FL[i]_{Nf_i}$, where $Nf_i = |FL[i]|$, represent, respectively, the cheapest and the fastest partial routes arriving at i in the the predefined sequence of nodes.

All these labels are computed in the initialization step of the ALNS procedure (Lines 1 and 2 in **Algorithm 5.1**). Then, they are maintained and updated during the search.

Using this data structure, removal and insertion operations performed by the move generation function (Line 5 in **Algorithm 5.1**) are efficiently handled. Indeed, to evaluate, for example, the impact of removing a node v_k from a sequence of nodes S, we do not have to compute the route cost associated with the new sequence. The removal saving is obtained by just examining labels associated with nodes v_{k-1} and v_{k+1}. In the following, we detail how elementary operations are evaluated and performed.

Algorithm 5.2 Forward labelling Algorithm

Input : $G_S = (V_S, E_S)$
Output : Vector of lists of forward labels FL
Initialize $FL[v_0] \leftarrow (0, 0, 0, v_0)$ forward label
for $k = 0$ **to** p **do**
  for all labels $L = (c, t, q, v_k) \in FL[v_k]$ **do**
    for all arcs $a = (v_k, v_{k+1}) \in A_S$ **do**
      if $t' = t + st_{v_k} + t_a \leq l_{v_{k+1}}$ **then**
        $L' = (c + c_a, max\{t', e_{v_{k+1}}\}, q + d_{v_{k+1}}, v_{k+1})$
        if L' is not dominated by any label in $FL[v_{k+1}]$ **then**
          $FL[v_{k+1}] \leftarrow FL[v_{k+1}] \cup \{L'\}$
        end if
        if a label $L'' \in FL[v_{k+1}]$ is dominated by L' **then**
          $FL[v_{k+1}] \leftarrow FL[v_{k+1}] \setminus \{L''\}$
        end if
      end if
    end for
  end for
end for
return FL

Algorithm 5.3 Backward labelling Algorithm

Input : $G_S = (V_S, E_S)$
Output : Vector of lists of backward labels BL
Initialize $BL[v_{p+1}] \leftarrow (0, l_{v_{p+1}}, 0, v_{p+1})$ label
for $k = p$ **downto** 0 **do**
  for all label $L = (c, t, q, v_{k+1}) \in BL[v_{k+1}]$ **do**
    for all arc $a = (v_k, v_{k+1}) \in A_S$ **do**
      if $t' = t - t_a - st_{v_k} \geq e_{v_k}$ **then**
        $L' = (c + c_a, min\{t', l_{v_k}\}, q + d_{v_k}, v_k)$
        if L' is not dominated by any label in $BL[v_k]$ **then**
          $BL[v_k] \leftarrow BL[v_k] \cup \{L'\}$
        end if
        if A label $L'' \in BL[v_k]$ is dominated by L' **then**
          $BL[v_k] \leftarrow BL[v_k] \setminus \{L''\}$
        end if
      end if
    end for
  end for
end for
return BL

5.4.1.1 Removal operation

For a removal operation, we need to evaluate the saving in route cost and once a candidate move is selected, we need to update labels for nodes in the modified sequence.

The removal saving of a customer v_k from the route defined by the sequence S is given by the difference between the cost associated with $R(S)$ and the cost associated with route $R(S \setminus \{v_k\})$. Using our data structure, this can be computed as follows:

1. Extend forward labels in $FL[v_{k-1}]$ along parallel arcs $(v_{k-1}, v_{k+1})^p$ and store obtained labels in a new list $FL'_{v_{k+1}}$ in the non-decreasing order of travel costs;
2. From labels in $FL'_{v_{k+1}}$ and in $BL[v_{k+1}]$ select the combination (L_f, L_b) with $L_f \in FL'_{v_{k+1}}$, $L_b \in BL[v_{k+1}]$ and $t(L_f) \leq t(L_b)$ that minimizes $c(L_f) + c(L_b)$;
3. Return $c(R(S)) - (c(L_f) + c(L_b))$ as removal saving for customer v_k

Once a customer v_k is selected to be removed, we have to update forward labels and backward labels. For the modified sequence, forward labels arriving at nodes v_i; $i = 1, ..., k-1$ remain unchanged. Forward labels at v_{k+1} are updated using labels in $FL_{v_{k+1}}$ and forward labels at nodes v_i; $i = k+2, ..., p+1$ are recomputed using **Algorithm 5.2**. Similarly, only backward labels arriving at nodes v_i; $i = k-1, ..., 0$ are recomputed using **Algorithm 5.3**.

5.4.1.2 Insertion operation

To evaluate the insertion cost of a customer u in a route $R(S)$ between two customers v_k and v_{k+1}, we have, first, to ensure the feasibility of such insertion with respect to capacity and time window constraints. Capacity constraints could be checked in a more global level, the insertion of u in any position in $R(S)$ is feasible in term of capacity if $d(S) + d_u \leq Q$ where $d(S) = \sum_{v \in S} d_v$. In this case, the insertion of customer u in a position v_k (between customers v_k and $v_k + 1$) is feasible if there is a partial route that visits u immediately after v_k, arrives at u not later than l_u then reaches v_{k+1} at time $t \leq BL[v_{k+1}]_{Nb_{v_{k+1}}}$; $Nb_{v_{k+1}} = |BL[v_{k+1}]|$.

The insertion cost of a customer u in a route $R(S)$ at position v_k is computed as follows:

1. Extend forward labels in $FL[v_k]$ along parallel arcs $(v_k, u)^p$ and store obtained labels in a new list FL'_u in a non-decreasing order of travel costs;
2. Extend backward labels in $BL[v_{k+1}]$ along parallel arcs $(u, v_{k+1})^p$ and store obtained labels in BL'_u in the non-decreasing order of travel costs;
3. From labels in FL'_u and in BL'_u select the combination (L_f, L_b); $L_f \in FL'_u$ and $L_b \in BL'_u$ such that $t(L_f) \leq t(L_b)$ that minimizes $c(L_f) + c(L_b)$;

4. Return $(c(L_f)+c(L_b))-c(R(S))$ as insertion cost of customer u in route $R(S)$ at position v_k

As in the removal operation, once an insertion of a customer u in a route $R(S)$ at a position v_k has to be performed, we have to update only a subset of forward and backward labels for the modified sequence $S' = (v_0, ..., v_k, u, v_{k+1}, ..., v_{p+1})$. The procedure updates forward labels in $FL[u]$ using labels in FL'_u and recomputes only forward labels arriving at nodes v_i $i = k+1, ..., p+1$, using **Algorithm 5.2**. Similarly, backward labels in $BL[u]$ are updated using BL'_u and only backward labels arriving at nodes v_i $i = k, ..., 0$ are recomputed using **Algorithm 5.3**.

5.4.2 Initial Solution

To provide an initial solution to our heuristic, we propose to adapt Clark and Wright's *Savings Algorithm*. It is a heuristic algorithm introduced by Clarke and Wright [20] for the classical vehicle routing problem. This algorithm is based on the so-called savings concept that consists in expressing the cost saving obtained by concatenating two routes into one feasible route.

Consider two nodes i and j served, initially, in two different routes such that i is the last node served in its route that we denote by $R(i)$ and j is the first to be served in $R(j)$. By merging routes $R(i)$ and $R(j)$ such that j is visited immediately after i, the occurred saving sav_{ij} is given by:

$$sav_{ij} = c_{(i,0)} + c_{(0,j)} - c_{(i,j)} \tag{5.11}$$

The higher is the value of sav_{ij}, the more attractive is to visit points i and j in the same route.

With the multigraph setting, the saving associated with a pair of nodes (i, j) and induced by merging $R(i)$ and $R(j)$ depends on the choice of three arcs: the arc $(i, 0)^x$ connecting i the last customer in $R(i)$ to the depot location 0, the arc $(0, j)^y$ connecting 0 to j the first customer visited in $R(j)$ and the arc $(i, j)^z$ used to link i and j in the new route. For this, we define a savings as follows:

$$sav_{ij}^{x,y,z} = c_{(i,0)^x} + c_{(0,j)^y} - c_{(i,j)^z} \tag{5.12}$$

Note that for a saving $sav_{ij}^{x,y,z}$, routes $R(i)$ and $R(j)$ are concatenated only if the following conditions are satisfied:

1. This can be done without modifying an already established connection between two customers;

2. The obtained route does not violate capacity and time windows constraints.

The first condition ensures that decisions made so far cannot be reconsidered. These decisions correspond to services sequencing and the selection of arcs used to link consecutive customers. Consequently, when constructing a route R by concatenating $R(i)$ and $R(j)$ we don't have to recompute the sequence of arcs linking customers in R. This sequence of arcs is given by $\{(u,v) \in A; (u,v) \in R(i), v \neq 0\} \cup \{(i,j)\} \cup \{(u,v) \in A; (u,v) \in R(j), u \neq 0\}$. The second condition indicates that only feasible routes are accepted. For a concatenation associated with a savings $sav_{ij}^{x,y,z}$, we need to make sure that time window constraints are respected for all customers initially in route $R(j)$. For this, we use the data structure proposed in Section 5.4.1. Due to the first condition, only one forward label and one backward label are associated with each customer in the solution. In this case, the concatenation associated with a savings $sav_{ij}^{x,y,z}$ is feasible if $t(FL[i]) + t_{(i,j)^z} \leq t(BL[j])$ and $q(FL[i]) + q(BL[j]) \leq Q$.

The scheme of the *Adapted Savings Algorithm* is described in **Algorithm 5.4**.

Algorithm 5.4 Multigraph Savings Algorithm

1: **Input :** $G = (V, A)$ a multigraph
2: **Output :** a solution s
3: **for all** $i \in V/\{0\}$ **do**
4: Assign i to a route $R(i)$
5: $s \leftarrow s \cup \{R(i)\}$
6: **end for**
7: Compute forward and backward labels associated with each route
8: **for all** pair of nodes (i, j) **do**
9: **for all** (x, y, z) such that $(i, 0)^x \in R(i)$, $(0, j)^y \in R(j)$ and $(i, j)^z \in A$ **do**
10: $sav_{ij}^{x,y,z} = c_{(i,0)^x} + c_{(0,j)^y} - c_{(i,j)^z}$ // Compute savings
11: **end for**
12: **end for**
13: Sort the list of savings in the descending order
14: **for all** Savings $sav_{ij}^{x,y,z}$ **do**
15: **if** i is the last node visited in $R(i)$ and j is the first node visited in $R(j)$ **then**
16: **if** $t(FL[i]) + t_{(i,j)^z} \leq t(BL[j])$ and $q(FL[i]) + q(BL[j]) \leq Q$ **then**
17: Combine $R(i)$ and $R(j)$ in a new route $r = R(i) \oplus R(j)$
18: Update forward labels for all customers $u \in R(j)$
19: Update backward labels for all customers $v \in R(i)$
20: $s \leftarrow s \cup \{r\}/\{R(i), R(j)\}$
21: **end if**
22: **end if**
23: **end for**
24: **return** s

In the general scheme of the ALNS heuristic, we apply **Algorithm 5.4** to construct an initial solution s. Then in the initialization step (line 2 in **Algorithm 5.1**), we update our data

structures (forward and backward labels) for all node sequences in s using **Algorithms 5.2** and **5.3**.

5.4.3 Removal Heuristics

The destroy function in the move generation procedure starts by selecting the customers to be removed. These customers are selected according to a cost that depends on the evaluation strategy. We propose three removal heuristics based on different evaluation strategies. Recall that for all heuristics removing operations are evaluated using the data structure and the procedure described in Section 5.4.1.1. Once customers to be removed are selected, forward and backward labels associated with current solution have to be updated.

A removal heuristic takes as inputs a feasible solution s and a number q of customers to be removed, and returns a set of routes and a set of q removed customers.

5.4.3.1 Adapted Shaw removal heuristic

This heuristic was first proposed by Shaw [124] for the VRPTW and next adapted by Ropke and Pisinger [118] for the PDPTW. The main idea is to remove "similar" costumers, so that we may obtain new, perhaps, better solutions after the reinsertion. Due to the tight structure of VRPTW solutions, removing very different customers might force the repair function to reinsert these customers at their original positions.

For a given solution, the similarity of two customers i and j is evaluated using a relatedness measure $R(i, j)$. The lower $R(i, j)$ is, the more related are the two customers. $R(i, j)$ is defined as follows:

$$R(i,j) = \varphi c_{(ij)^0} + \omega|t_i - t_j| + \alpha|d_i - d_j| + \beta(1 - \frac{|RC_i \cap RC_j|}{min\{|RC_i|, |RC_j|\}}) + X_{ij} \tag{5.13}$$

where $c_{(i,j)^0}$ denotes the cost of the less costly arc linking i to j, t_i denotes the service starting time at customer i, d_i is the demand of customer i, RC_i represents the set of positions where i can be inserted in the current solution and $X_{ij} \in \{0, 1\}$ indicates if i and j are served by the same vehicle. φ, ω, α and β represent weights associated to different terms and are in $[0, 1]$. The term weighted by φ measures the cost, the term weighted by ω measures the degree of the similarity in service time, the term weighted by α evaluate the similarity in demand and the term weighted by β measures the relatedness in the possible deviation in the current solution to reinsert i and j.

It is clear that except for the term weighted by β, all terms in $R(i, j)$ are directly given from the problem data (for $c_{(ij)^0}$ and d_i) or from the solution structure (for t_i and X_{ij}). The

main purpose of introducing the term weighted by β is to select customers with "similar flexibility" and thus the repair function would be able to find a different solution, probably, improving the current solution. In the other hand, computing RC_i for every customer i could be time consuming as we need to check all possible insertion positions for i in each route. A sensitivity analysis of the impact of this term on the heuristic performance is proposed in Section 5.5.4.

The heuristic starts by selecting, randomly, a customer to remove. Then, it selects subsequently the most similar customer to the already removed ones. We introduce some randomness in the selection of customers to be removed using a parameter p_s. The procedure is described in **Algorithm 5.5**.

Algorithm 5.5 Adapted Shaw Removal Heuristic

Input : solution s and $q \in \mathbb{N}$
Parameter : $p_s \in \mathbb{R}_+$
Output : set of customers to be removed $\mathfrak{R} = \emptyset$
Randomly select a customer c and remove it from solution s
$\mathfrak{R} = \{c\}$
while $|\mathfrak{R}| < q$ **do**
    Randomly select a customer $c \in \mathfrak{R}$
    Sort customers $i \in s \setminus \mathfrak{R}$ in an array $Cust$ according to decreasing $R(c, Cust[i])$ values
    Choose a random value $y \in [0, 1)$
    Select the i^{th} customer in $Cust$ where $i = \lfloor y^{p_s} * |Cust| \rfloor$
    $\mathfrak{R} = \mathfrak{R} \cup \{Cust[i]\}$
end while
return $\mathfrak{R}$

5.4.3.2 Random Removal Heuristic

As in [118], this removal heuristic simply selects q customers in a random way and removes them from the solution.

5.4.3.3 Worst Removal Heuristic

It was introduced by Ropke and Pisinger [118]. Basically, it tries to remove the misplaced customers in the current solution. To do this, we define the *marginal cost* of serving a customer i in a solution s as $cost(i, s) = f(s) - f_{-i}(s)$ where $f(s)$ and $f_{-i}(s)$ are solution costs before and after removing customer i from solution s. In order to rearrange the solution and obtain a better solution cost, we remove customers with the highest marginal costs $cost(i, s)$

and reinsert them in different positions.

As in the **Show Removal heuristic**, we introduce some randomness in the selection of the customer to remove using a parameter p_w.

The worst Removal heuristic proceeds as follows:

Algorithm 5.6 Worst Removal Heuristic

Input : solution s and $q \in \mathbb{N}$
Parameter : $p_w \in \mathbb{R}_+$
Output : set of customers to be removed $\mathfrak{R} = \emptyset$
Compute $cost(i, s) = f(s) - f_{-i}(s)\ \forall i \in V \setminus \{0\}$
$\mathfrak{R} = \emptyset$
while $|\mathfrak{R}| < q$ **do**
    Sort customers $i \in s \setminus \mathfrak{R}$ in an array $Cust$ according to decreasing $cost(i, s)$
    Choose a random value $y \in [0, 1)$
    Select the i^{th} customer in $Cust$ where $i = \lfloor y^{p_w} * |Cust| \rfloor$
    $\mathfrak{R} = \mathfrak{R} \cup \{Cust[i]\}$
    Update $cost(i, s)\ \forall i \in s \setminus \mathfrak{R}$
end while
return $\mathfrak{R}$

5.4.4 Insertion Heuristics

The repair function consists in rescheduling the removed customers into the fixed routes. In our search scheme, we use four different insertion strategies: two strategies are proposed by Ropke and Pisinger [118] and are updated to handle the multigraph setting (5.4.4.1 and 5.4.4.2), a simple strategy used to diversify the search procedure (5.4.4.3) and a non-myopic heuristic based on a look-ahead strategy (5.4.4.4).

Note that all presented heuristics perform on q iterations: at each iteration a customer is selected according to a cost that is defined depending on the insertion strategy. Typically, insertion costs are computed in preprocessing and then updated during the insertion procedure. To do this, we use the data structure and procedure described in Section 5.4.1. At each iteration, a customer is inserted. Thus a new route is constructed and forward and backward labels are updated as described in Section 5.4.1.2.

5.4.4.1 Greedy Insertion heuristic

It is a gluttonous construction heuristic. At each iteration, the heuristic selects the customer to insert as follows:

1. For each customer $i \in \mathfrak{R}$ and for each route $k \in \mathcal{K}$, we define $\Delta f_{i,k}$ as the cost induced by inserting customer i into route k in its best position.
2. For each customer i, we define the insertion cost c_i as the change in the solution cost when $i \in \mathfrak{R}$ is inserted at its best position over all routes, i.e. $c_i = \min_{k \in \mathcal{K}}\{\Delta f_{i,k}\}$.
3. The customer i^* that minimizes *insertion cost* is selected and inserted it at its best position,

$$c_{i^*} = \min_{i \in \mathfrak{R}}\{c_i\} \tag{5.14}$$

Note that, to compute $\Delta f_{i,k}$ for a customer $i \in \mathfrak{R}$, we have to evaluate the insertion cost for every possible insertion position of i in route k. Using algorithms proposed in [61] and in [85], a new sequence of arcs have to be computed for every possible position which could be time consuming. This issue is efficiently handled by our procedure.

In each iteration (1-3), only one route is changed. So, we do not have to recalculate insertion costs in other routes, and only insertions in the modified route are evaluated in next iterations.

5.4.4.2 Regret insertion heuristic

The main idea of this insertion heuristic is to improve the basic greedy heuristic by integrating a sort of look-ahead information in the process of selecting the customer to insert at each iteration. This is performed by evaluating the insertion of a costumer $i \in \mathfrak{R}$ as the difference between the cost of planning i in its best route $k_1 \in \mathcal{K}$ and the cost of planning i in its second best route $k_2 \in K$. Let a_{ik} be an integer value such that $1 \leq a_{ik} \leq |\mathcal{K}|$ that denotes the route for which the customer i has the k^{th} lowest insertion cost. Thus, the cost of the best insertion for a customer i defined in the greedy heuristic is given by $c_i = \Delta f_{ia_{i1}}$. The *regret value* is, then, given by $r_i = \Delta f_{ia_{i2}} - \Delta f_{ia_{i1}}$. This value measures, at each iteration, the regret of not inserting a customer i into its best route and inserting it in its second-best route. So, in each iteration, the regret heuristic selects the customer i^* that maximizes the regret value

$$r_{i^*} = \max_{i \in \mathfrak{R}} r_i \tag{5.15}$$

5.4.4.3 Simple insertion heuristic

It is a simple construction heuristic that assigns removed customers to new routes in a greedy way with respect to the fleet size. The insertion procedure is shown in Algorithm 5.7.

Algorithm 5.7 Simple Insertion Heuristic

Input : a set of removed customers $\mathfrak{R}$ and a set of fixed routes $FR = \{r\ route\ ; r \in s \setminus \mathfrak{R}\}$

Compute $cost(i, s) = f(s) - f_{-i}(s)\ \forall i \in V \setminus \{0\}$
set of new routes $NR = \emptyset$
while $|\mathfrak{R}| > 0$ **do**
  Select a customer $i \in \mathfrak{R}$
  if $|FR| + |NR| < |\mathcal{K}|$ **then**
    Assign i to a new route r
    $NR = NR \cup \{r\}$
  else
    Select a route $r \in NR$
    Assign i to a r
  end if
  $\mathfrak{R} = \mathfrak{R} \setminus \{i\}$
end while
Construct a new solution $s_{new} = FR \cup NR$
return s_{new}

The aim of this heuristic is to generate new solutions which, probably, doesn't improve the current solution but may prompt the search procedure to explore different areas from the solution space. In addition, it could be more interesting to serve a customer in an independent route in some cases.

5.4.4.4 Non-myopic insertion heuristic

This heuristic consists in defining the cost of an insertion as the sum of its *"local cost"* (the change in the objective function value by inserting the customer in its best position over all routes) and a charge that estimates the impact of its insertion on other insertions that would be performed in the next iterations.

Let c_i denotes the insertion cost of customer $i \in \mathfrak{R}$ in the current solution s computed as in the **greedy insertion heuristic** and let c^i_j denotes the insertion cost of customer $j \in \mathfrak{R}$, $j \neq i$ in the new solution s^i obtained after inserting i at its best position. In the non-myopic insertion heuristic, we define the *impact value* c^*_i of a customer i as $c^*_i = c_i + \sum_{j \in \mathfrak{R}; j \neq i} (c^i_j - c_j)$. For a non already inserted customer j the quantity $c^i_j - c_j$ measures how the insertion of i could impact the insertion of j. In each iteration, the insertion procedure selects the customer that minimizes c^*_i and insert it at its minimum cost position.

Note that in order to determine the *impact value* of a customer i, we only change one route r and we have to evaluate only change in cost $\Delta f_{j,r}$ induced by inserting customer $j \in \Re$ in r; if $\Delta f_{j,r} < c_j$, the insertion of i in r results in a best insertion cost for j in a subsequent iteration and the *impact value* for i is reduced by $c_j - \Delta f_{j,r}$, but if $\Delta f_{j,r} \geq c_j$ the insertion of i in r has an impact on the insertion cost for j equals to $c_j - \Delta f_{j,r} \geq 0$.

Contrary to the ***greedy insertion heuristic*** that inserts at each iteration the customer with the lowest *"local insertion cost"* c_i regardless other insertions, the **non-myopic heuristic** may prefer to insert a customer j with a higher insertion cost $c_j \geq c_i$ but with a lower impact on next insertions $\sum_{k \in \Re; k \neq j} (c_k^j - c_k) \leq \sum_{k \in \Re; k \neq i} (c_k^i - c_k)$ such that $c_j^* \leq c_i^*$.

One could expect that this insertion strategy would be time consuming as a forecast is performed each time a customer insertion is evaluated. On the other hand, such method would have an interesting impact on the search scheme as it may anticipate "bad" insertion operations that would tighten the solution structure and constrict subsequent insertions. To investigate these issues, some sensitivity analysis are presented in Section 5.5.4.

5.4.5 Adaptive Strategy for the control of the Removal/Insertion operators

In Sections 5.4.3 and 5.4.4, we defined three removal heuristics and four insertion heuristics that can be used in the search procedure. Naturally, different types of instances and even different solutions for the same instance can be handled by different combinations of removal and insertion heuristics. The success of the method depends on when to use each heuristic, but it is difficult to determine a priori which removal and insertion strategies would be more efficient for a given instance. For this purpose, we propose to use a control strategy introduced by Ropke and Pisinger [118] that defines which heuristic to be used at each step of the search procedure in an adaptive way.

The algorithm assigns a weight w_j; $j = 1, ..., 6$ to each heuristic and periodically adjusts them during the search. These weights reflect the success of every heuristic for earlier iterations. The selection of removal and insertion heuristics is made using a roulette wheel principle.

The selection mechanism proceeds as follows:

1. The entire search is divided into a number of segments. A segment is a number of iterations;

2. All weights are, equally, initialized at the beginning of the search scheme;

3. For each heuristic, the algorithm defines a score which measures how well the heuristic has performed along the last segment. The scores of all heuristics are set to zero at the beginning of each segment. At each iteration, we increase the scores of the applied removal and insertion heuristics by:

   - μ_1 : if the remove-insert operation results in a new global best solution;
   - μ_2 : if the obtained solution is accepted and improves the current solution;
   - μ_3 : if the obtained solution is accepted but has a total cost worse than the one of the current solution.

4. At the end of each segment, the algorithm computes new weights that are used to select the heuristics in the next segment of the search. Let $w_{i,j}$ be the weight of a heuristic i used in the last segment j. The new weight $w_{i,j+1}$ of the heuristic i is computed by:

$$w_{i,j+1} = w_{i,j} * (1 - r) + r * \frac{\pi_i}{\eta_i} \tag{5.16}$$

   where π_i is the total score of the heuristic i computed in the segment j, η_i represents the number of times heuristic i is applied during segment j and r is a *reaction factor* which controls how the adjustment algorithm reacts to the effectiveness of the heuristic during the last segment.

Note that, the computed scores does not only evaluate the attitude of heuristics to improve a solution but also enhance the heuristics that can diversify the search, and these are rewarded by μ_3.

5.4.6 Acceptance criteria

If the ALNS heuristic accepts only solutions improving the current solution, it could, probably, get trapped into local optima. To avoid such a situation, we use an acceptance criteria inspired from the simulated annealing metaheuristic. It consists in accepting a solution s' with a probability given by $exp(\frac{-(s'-s)}{T})$ where s is the current solution and T is the temperature. The temperature starts at T_{start} which is fixed in such a way that the first accepted solution with a probability equal to 0.5 is 5% worse than the initial solution. Then the temperature is decreased every iteration with $T = T * c$ where $0 < c < 1$ is the *cooling rate*. Here we fix a limit for the number of iterations to stop the execution of the algorithm.

5.5 Computational experiments

In this section, we describe our experimental computations. First, we present the benchmark instances in Section 5.5.1. In Section 5.5.2, we describe how ALNS parameters are tuned.

In Section 5.5.3 we evaluate the performance of our ALNS heuristic and the impact of the multigraph representation on the solution quality. In section 5.5.4, we present sensitive analysis on impact of introduced components on the performance of the method.

All presented algorithms are implemented in the C++ programming language. Tests are run on an Intel CORE i5 2.6 GHz computer with 4GB of memory.

5.5.1 Test Data

In our experiments, we used four sets of VRPTW instances provided by Ben Ticha et al. [8];

- A first set consists of 90 instances derived from the Solomon [126] benchmark instances; 45 instances with 25 customers and 45 instances with 50 customers. To generate multigraph instances, Ben Ticha et al. [8] assigned to each arc in the original instance two attributes; a travel cost defined by the Euclidean distance and a travel time computed using a correlation rule according to cost. Three correlation degrees are used: no-correlation (NC), weak correlation (WC) and strong correlation (SC).
- A second set of 30 instances is provided by Letchford et al. [89]. It is generated from sparse graphs that simulate urban road networks. These graphs are with $N \in \{25, 50, 75, 100\}$ nodes and each is given with a probability $p = 0.66$ to be a customer. Arcs are defined with a travel cost given by the Euclidean distance and a travel time computed using a correlation rule according to cost. Instances with three different levels of correlation are generated (NC, WC and SC).
- A third set of 45 instances are based on road networks generated by Ben Ticha et al. [8] using the same procedure described in [89]. These instances are subdivided into three classes of 15 instances with, respectively, 25 customers out of 100 nodes, 50 customers out of 100 nodes and 50 customers out of 200 nodes. For each 5 instances of a class, travel times are non-correlated, weakly correlated or strongly correlated to costs.
- A fourth set of 12 instances are generated based on real spatial data of two road networks in the region of *Provence-Alpes-Cote d'Azur* in the south of France. In each road network, an arc represents a road segment and is defined by a length, a maximum allowed speed and a travel direction. Travel times are computed using road segment speeds and lengths. 6 instances are generated from each road network; 2 instances with 25 customers, 2 instances with 50 customers and 2 instances with 75 customers.

In all cases, parallel arcs between a pair of customers represent the set of non-dominated bi-criteria shortest paths. For more details on instance characteristics, interested readers are referred to [8].

Table 5.1: *Parameters values*

Operator	Parameter	Selected value
Shaw removal	Weight associated with cost term: ϕ	4
	Weight associated with service time term: ω	5
	Weight associated with demand term: α	3
	Weight associated with deviation term: β	10
	Randomness degree: p_s	6
Worst removal	Randomness degree: p_w	5
Weight adjustment algorithm	Initial weight	100
	Gain for a new global best solution: μ_1	500
	Gain for an improving solution: μ_2	200
	Gain for a non-improving solution: μ_2	150
	Reaction factor r	0.1
Acceptance method	Cooling rate c	0.99975

5.5.2 Parameters tuning

A first set of experiments are performed to adjust the parameters of our heuristics. To do this, a set of representative tuning instances is selected and on which different combination of parameters are tested.

We use the following strategy for tuning all parameters: First, we examine the parameters that control the **Shaw removal heuristics** and the **Worst removal heuristic**. Both heuristics are tested separately with the **Greedy insertion heuristic** where each time a parameter is allowed to take a number of values, while the rest of the parameters are kept fixed. For each parameter setting, the solution method is applied five times on tuning instances, and the setting with the best average solution quality is selected. Next, we examine parameters associated with the weight adjustment algorithm and the acceptance method. These parameters are gains μ_1, μ_2 and μ_3, the *reaction factor r* and the *cooling rate c*. We apply the complete search scheme and each time a parameter is adjusted. Finally, we examine the impact of the number of customers q to rearrange in each solution.

Table 5.1 summarizes the parameter values for which the heuristic shows the best average behavior with the tuning instances.

5.5.3 Computational results

In this section, we perform a set of experiments to evaluate the performance of our solution method. First, we compare solutions obtained with the Adapted Savings and the ALNS algorithms to optimal solutions with the multigraph representation. These optimal solutions are provided in [8] and computed using a branch-and-price algorithm for the multigraph based VRPTW. Then, we investigate the impact of the multigraph representation on the solution

quality. To do this, we compare solutions obtained with the ALNS to optimal solutions computed using a branch-and-price algorithm on two customer-based graphs: a *min-cost graph* that consists of less costly paths and a *min-time graph* that consists of fastest paths. For each instance, the ALNS heuristic is applied 10 times and the best solution cost found is reported. Computing times for the branch-and-price algorithms are limited to 7,500 seconds.

We report the obtained results for the adapted instances of Solomon [126] in Tables 5.2 and 5.3, for Letchford et al. [89] instances in Table 5.4, for Letchford et al.-like (LL) instances in Table 5.5 and for real instances in Table 5.6. Column "Opt" indicates computing times for optimal solution in the multigraph. Column "Adapted Savings" reports computing time in seconds (column "CPU") and gap between the Savings solution and the optimal solution in the multigraph (column "Gap MG"). Results for the ALNS heuristic are presented in column "ALNS"; columns "Gap MG", "Gap MC" and "Gap MT" report the gap between the ALNS solution and optimal solutions with, respectively, multigraph, min-cost graph and min-time graph representations. The gap is computed as follows:

$$Gap(\%) = \frac{solution\ cost\ with\ ALNS\ -\ optimal\ solution\ cost}{optimal\ solution\ cost} * 100\% \tag{5.17}$$

In Table 5.4, the first column "Instance" shows instance name which indicates the number of nodes in the original road network, the number of customers in the multigraph and whether time windows are wide (WTW) or narrow (NTW). In all Tables, column "Corr" reports the correlation level. In Table 5.5 columns "# nodes" and "# cust" indicate, respectively, the number of nodes in the original road network and the number of considered customers. Column "Inst" specify the instance index. In Table 5.6, "Zone" indicates the used real road network.

5.5.3.1 Savings solution

The adapted savings algorithm is used to provide an initial solution for the search scheme. For this reason, the computing time is an important criterion to evaluate the efficiency of the proposed algorithm as well as the solution quality.

From Tables 5.2 and 5.3, it comes out that the adapted savings algorithm provides a solution within 0.004 and 0.010 seconds in average for the adapted Solomon [126] instances with respectively 25 and 50 customers. The average gap between the savings solution and the optimal solution is almost 35% for instances with with 25 customers and is 34% for instances with 50 customers.

Table 5.4 shows that for Letchord et al. [89] instances, the adapted savings algorithm provides a solution within 0.007 seconds in average. Obtained solutions are in average 26.9% worst than optimal solutions.

Table 5.2: *Results for adapted Solomon [126] instances with 25 customers*

		Opt	Adapted Savings		ALNS			
Corr	Instance	CPU	CPU	Gap MG(%)	CPU	Gap MG (%)	Gap MC (%)	Gap MT (%)
NC	r101	0.6	0.005	17.0	8.4	**0.0**	0.0	**-46.1**
	r102	1.7	0.004	28.0	10.7	**0.0**	**-1.0**	**-27.3**
	r103	12.7	0.004	58.2	11.8	**0.0**	0.0	**-32.5**
	r104	31.0	0.004	24.0	15.4	**0.0**	0.0	**-25.0**
	r105	2.3	0.007	30.9	10.3	**0.0**	**-1.6**	**-35.9**
	c101	7500.0	0.005	–	13.8	–	**-0.1**	**-57.2**
	c102	7500.0	0.007	–	22.6	–	–	**-52.7**
	c103	2197.0	0.008	86.6	26.2	**0.0**	**-9.7**	**-44.6**
	c104	7500.0	0.009	–	42.7	–	0.0	**-44.1**
	c105	54.6	0.007	79.3	19.7	**0.0**	**-3.4**	**-53.8**
	rc101	0.7	0.009	41.2	7.5	**0.0**	**-9.3**	**-41.1**
	rc102	10.9	0.007	63.7	12.1	**0.0**	**-11.2**	**-34.9**
	rc103	613.9	0.008	63.2	15.3	**0.0**	**-2.2**	**-33.7**
	rc104	2728.0	0.005	51.5	17.8	**0.0**	**-4.9**	**-20.1**
	rc105	9.8	0.005	44.4	8.4	**0.0**	**-3.6**	**-27.9**
WC	r101	0.2	0.002	17.3	7.6	**0.0**	0.0	**-9.8**
	r102	1.2	0.002	6.8	6.3	**0.0**	0.0	**-6.1**
	r103	2.3	0.003	15.5	6.1	**0.0**	0.0	**-5.6**
	r104	4.7	0.003	20.6	6.9	**0.0**	0.0	**-3.7**
	r105	0.9	0.002	10.9	6.0	**0.0**	0.0	**-10.2**
	c101	206.5	0.004	37.7	4.9	**0.0**	**-4.6**	–
	c102	7500.0	0.004	–	10.5	–	–	**-12.7**
	c103	480.4	0.003	50.4	13.4	**0.0**	0.0	**-20.9**
	c104	7500.0	0.003	–	9.8	–	0.0	**-6.2**
	c105	27.3	0.003	25.1	6.2	**0.0**	0.0	**-23.2**
	rc101	7.7	0.004	25.6	5.0	**0.0**	**-7.9**	**-10.4**
	rc102	983.5	0.004	38.3	7.1	**0.0**	**-12.0**	**-3.8**
	rc103	713.3	0.003	64.1	7.8	**0.0**	**-2.4**	**-3.3**
	rc104	835.0	0.004	47.2	8.1	**0.0**	**-0.2**	**-2.6**
	rc105	2.8	0.003	30.5	5.6	**0.0**	**-1.5**	**-7.7**
SC	r101	0.2	0.002	4.7	7.7	**0.0**	0.0	0.0
	r102	0.5	0.002	8.4	6.1	**0.0**	0.0	**-1.1**
	r103	0.9	0.002	1.8	4.4	**0.0**	**-1.8**	0.0
	r104	4.2	0.002	15.1	4.6	**0.0**	0.0	**-0.6**
	r105	0.8	0.002	6.8	4.7	**0.0**	0.0	**-0.1**
	c101	6.4	0.003	24.2	3.3	**0.0**	0.0	**-1.7**
	c102	5.5	0.003	18.2	3.9	**0.0**	0.0	0.0
	c103	96.4	0.003	41.2	4.7	**0.0**	0.0	**-1.1**
	c104	1717.6	0.002	47.9	5.3	**0.0**	0.0	0.0
	c105	0.5	0.003	36.2	3.3	**0.0**	0.0	0.0
	rc101	13.9	0.002	44.7	4.5	**0.0**	**-4.7**	**-0.8**
	rc102	397.7	0.003	34.8	4.2	**0.0**	0.0	**0.0**
	rc103	5.9	0.003	58.2	4.1	**0.0**	0.0	**-0.1**
	rc104	13.1	0.003	21.7	4.4	**0.0**	0.0	0.0
	rc105	21.7	0.002	43.4	4.5	**0.0**	0.0	**-0.2**

NOTE *: – indicates that the algorithm has not terminated within 7,500 seconds*

Table 5.3: *Results for adapted Solomon [126] instances with 50 customers*

		Opt	Adapted Savings		ALNS			
Corr	Instance	CPU	CPU	Gap MG (%)	CPU	Gap MG (%)	Gap MC (%)	Gap MT (%)
NC	r101	17.6	0.011	12.1	27.9	1.1	0.7	**-41.4**
	r102	52.8	0.015	17.1	32.6	0.1	**-0.4**	**-31.3**
	r103	593.7	0.018	34.5	49.2	1.0	**-1.3**	–
	r104	6798.9	0.034	45	109.3	1.3	**-1.8**	–
	r105	26.9	0.025	39.7	33.4	1.8	1.3	**-42.0**
	c101	7500.0	0.016	–	29.8	–	–	–
	c102	7500.0	0.018	–	57.8	–	–	**-54.6**
	c103	7500.0	0.022	–	90.1	–	–	**-42.0**
	c104	7500.0	0.032	–	302.1	–	–	–
	c105	7500.0	0.016	–	48.1	–	–	**-58.4**
	rc101	108.0	0.018	54.8	25.6	0.2	**-12.9**	**-40.3**
	rc102	321.6	0.018	56.8	36.4	0.9	**-7.9**	**-33.0**
	rc103	6821.9	0.018	78.6	47.1	1.1	0.2	**-26.6**
	rc104	7500.0	0.02	–	76.7	–	–	–
	rc105	2358.4	0.02	66.1	32.2	0.9	**-7.8**	**-32.6**
WC	r101	1.2	0.003	18.9	10.4	0.3	**-3.0**	**-22.3**
	r102	6.3	0.005	22.9	12.7	0.6	**-1.1**	**-9.0**
	r103	65.4	0.006	32.9	17.0	0.3	**-1.3**	**-9.9**
	r104	1304.3	0.007	21.9	31.3	0.4	0.4	**-5.8**
	r105	17.0	0.005	12.4	11.8	0.5	0.0	**-8.2**
	c101	7500.0	0.005	–	14.4	–	–	–
	c102	7500.0	0.007	–	34.4	–	–	–
	c103	7500.0	0.008	–	46.1	–	–	–
	c104	7500.0	0.007	–	98.7	–	–	–
	c105	7500.0	0.006	–	24.3	–	**-3.8**	**-18.9**
	rc101	110.3	0.006	33.8	10.6	**0.0**	**-7.2**	**-7.3**
	rc102	7500.0	0.006	–	14.8	–	**-5.1**	–
	rc103	7500.0	0.006	–	18.6	–	**-3.3**	–
	rc104	7500.0	0.007	–	25.4	–	–	–
	rc105	68.4	0.008	49.1	14.0	**0.0**	**-8.8**	**-7.8**
SC	r101	1.0	0.003	16.5	8.5	0.1	**-1.1**	**-0.9**
	r102	6.8	0.003	18.8	9.3	**0.0**	0.0	**-0.7**
	r103	71.1	0.003	14.4	10.7	**0.0**	0.0	**-0.1**
	r104	7500.0	0.003	–	14.9	–	–	–
	r105	14.9	0.003	16.6	9.3	0.2	**-0.7**	**-0.1**
	c101	106.4	0.002	26.4	9.5	**0.0**	0.0	**-0.5**
	c102	53.8	0.003	65.7	14.2	**0.0**	0.0	**-1.9**
	c103	699.9	0.004	23.5	19.6	**0.0**	0.0	**-1.0**
	c104	7500.0	0.003	–	29.6	–	–	–
	c105	12.0	0.003	27.9	10.4	**0.0**	0.0	**-0.5**
	rc101	2385.9	0.003	48.1	8.8	**0.0**	0.0	**-1.0**
	rc102	7500.0	0.003	–	10.8	–	–	–
	rc103	7500.0	0.003	–	12.1	–	–	–
	rc104	7500.0	0.003	–	15.4	–	–	–
	rc105	6544.4	0.003	15.9	9.8	**0.0**	0.0	**-0.7**

NOTE *: – indicates that the algorithm has not terminated within 7,500 seconds*

Table 5.4: *Results for Letchford et al. [89] instances*

		Opt	Adapted Savings		ALNS			
Instance	Corr	CPU	CPU	Gap MG(%)	CPU	Gap MG (%)	Gap MC (%)	Gap MT (%)
25_16_NTW	NC	0.1	0.000	17.5	1.7	**0.0**	0.0	**-7.9**
25_16_WTW	NC	0.1	0.010	20.8	1.9	**0.0**	0.0	**-7.9**
25_16_NTW	WC	0.1	0.000	45.3	1.6	**0.0**	0.0	**-1.0**
25_16_WTW	WC	0.1	0.000	13.8	1.7	**0.0**	0.0	**-1.0**
25_16_NTW	SC	0.1	0.010	16.7	1.4	**0.0**	0.0	0.0
25_16_WTW	SC	0.1	0.000	16.7	1.6	**0.0**	0.0	0.0
50_33_NTW	NC	0.5	0.003	18.3	5.4	**0.0**	**-2.2**	**-2.6**
50_33_WTW	NC	398.0	0.000	21.9	6.4	**0.0**	0.0	**-1.1**
50_33_NTW	WC	1.8	0.000	23.6	5.9	**0.0**	0.0	**-0.1**
50_33_WTW	WC	7500.0	0.003	–	6.7	–	0.0	**-0.1**
50_33_NTW	WC	0.6	0.002	30.5	6.8	**0.0**	**	**-5.3**
50_33_WTW	WC	50.6	0.000	17.1	8.8	**0.0**	**	**-4.1**
50_33_NTW	SC	19.4	0.000	19.6	5.8	**0.0**	0.0	**-0.3**
50_33_WTW	SC	533.9	0.000	3.7	6.0	**0.0**	0.0	0.0
75_50_NTW	NC	0.7	0.000	20.7	13.7	**0.0**	**	**-8.4**
75_50_WTW	NC	152.3	0.000	18.2	16.4	**0.0**	**-2.5**	**-4.6**
75_50_NTW	WC	1.4	0.010	26.2	15.4	**0.0**	0.0	**-1.1**
75_50_WTW	WC	7500.0	0.010	–	18.3	–	**-0.1**	**-0.2**
75_50_NTW	WC	3.2	0.010	30.5	17.7	**0.0**	**-2.9**	**-11.8**
75_50_WTW	WC	353.5	0.010	40.5	22.9	1.7	0.0	**-7.7**
75_50_NTW	SC	1.1	0.040	9.8	11.1	**0.0**	0.0	**-0.1**
75_50_WTW	SC	5305.9	0.000	16.8	13.4	1.2	1.2	1.1
100_66_NTW	NC	64.7	0.030	53.6	24.5	**0.0**	**	**-7.2**
100_66_WTW	NC	550.1	0.020	39.8	26.8	1.8	**-3.7**	**-7.4**
100_66_NTW	WC	6.2	0.010	45.9	20.5	**0.0**	**	**-5.5**
100_66_WTW	WC	4391.6	0.010	44.3	25.7	**0.0**	**	**-6.6**
100_66_NTW	WC	13.0	0.012	57.1	25.2	**0.0**	**	**-7.9**
100_66_WTW	WC	593.6	0.020	31.7	31.5	1.0	**	**-7.1**
100_66_NTW	SC	4.5	0.000	24.9	18.5	**0.0**	0.0	**-0.9**
100_66_WTW	SC	7500.0	0.010	–	21.7	–	–	–

NOTE *: – indicates that the algorithm has not terminated within 7,500 seconds*

***: indicates that the instance is infeasible in the min-cost graph*

Table 5.5: *Results for LL instances*

				Opt	Adapted Savings		ALNS			
#nodes	#cust	Corr	Inst	CPU	CPU	Gap MG (%)	CPU	Gap MG (%)	Gap MC (%)	Gap MT (%)
100	25	NC	1	6.3	0.003	10.5	6.0	**0.0**	**-6.4**	**-3.5**
			2	1.4	0.005	27.2	5.8	**0.0**	0.0	**-10.6**
			3	5.7	0.007	26.0	7.3	**0.0**	**-2.3**	**-13.5**
			4	2.8	0.004	15.5	5.3	**0.0**	**-0.4**	**-5.7**
			5	2.5	0.006	20.0	9.2	**0.0**	**-1.1**	**-10.6**
		WC	1	2.0	0.002	9.1	4.0	**0.0**	0.0	**-3.3**
			2	71.5	0.003	14.3	5.9	**0.0**	**-1.7**	**-5.6**
			3	3.1	0.002	17.9	4.3	**0.0**	**-2.5**	**-0.4**
			4	0.8	0.004	1.1	3.9	**0.0**	0.0	**-2.7**
			5	10.4	0.003	15.3	4.7	**0.0**	**-3.1**	**-3.2**
		SC	1	21.1	0.002	8.9	2.9	**0.0**	0.0	**-0.4**
			2	1.3	0.002	12.4	3.2	**0.0**	0.0	**-0.2**
			3	9.7	0.003	16.4	4.6	**0.0**	0.0	**-2.1**
			4	0.6	0.002	42.5	3.4	**0.0**	0.0	0.0
			5	0.6	0.002	13.3	3.8	**0.0**	0.0	**-0.7**
	50	NC	1	242.0	0.012	29.9	18.6	0.7	**-1.3**	**-8.2**
			2	2296.5	0.013	32.7	26.1	**0.0**	**-4.7**	**-2.5**
			3	1045.6	0.015	34.5	25.2	0.1	**-4.0**	**-3.2**
			4	399.9	0.012	41.5	23.2	**0.0**	**-4.0**	**-7.0**
			5	82.2	0.012	26.7	20.0	**0.0**	**-2.7**	**-6.6**
		WC	1	104.8	0.008	14.6	14.6	**0.0**	0.0	**-5.4**
			2	245.5	0.008	22.5	13.3	0.2	**-2.9**	**-2.9**
			3	79.1	0.007	23.7	15.1	0.2	**-3.0**	**-1.7**
			4	42.2	0.006	14.6	14.2	**0.0**	0.0	**-4.7**
			5	291.3	0.006	41.9	13.8	**0.0**	0.0	**-2.0**
		SC	1	1120.1	0.003	23.4	12.0	1.0	0.6	1.0
			2	424.2	0.003	15.3	10.6	0.4	0.4	0.4
			3	21.4	0.003	19.1	10.8	0.1	**-3.9**	0.0
			4	44.9	0.003	18.8	12.0	**0.0**	0.0	**-0.5**
			5	5.4	0.002	14.6	10.1	**0.0**	0.0	**-0.4**
200	50	NC	1	1659,3	0,025	112,3	36,6	0,3	**-7,4**	**-5,9**
			2	191,3	0,022	55,1	28,8	**0,0**	**-4,8**	**-11,3**
			3	1045,1	0,017	109,8	24,0	0,3	**-3,7**	**-7,1**
			4	6775,5	0,023	96,2	41,3	0,5	**-5,1**	**-5,8**
			5	7500,0	0,026	133,8	34,1	—	**-3,9**	**-8,6**
		WC	1	4378,7	0,021	137,0	24,1	0,8	**-2,6**	**-5,4**
			2	181,7	0,013	61,7	18,3	**0,0**	**-3,5**	**-5,9**
			3	651,6	0,016	108,6	21,5	0,6	**-3,5**	**-4,6**
			4	695,4	0,012	91,6	27,2	**0,0**	**-3,9**	**-6,4**
			5	453,4	0,018	115,8	28,3	**0,0**	**-8,8**	**-8,3**
		SC	1	7500,0	0,006	77,5	14,5	—	**-3,7**	**-0,7**
			2	3892,8	0,006	73,7	14,6	0,2	0,2	**-0,8**
			3	203,3	0,003	54,0	11,4	0,1	0,1	**-0,2**
			4	81,8	0,003	56,0	10,6	0,2	0,2	**-0,4**
			5	67,5	0,004	25,9	10,4	**0,0**	0,0	**-0,2**

NOTE *: – indicates that the algorithm has not terminated within 7,500 seconds*

Table 5.6: *Results for Real instances*

			Opt	Adapted Savings		ALNS			
Zone	# cust	Inst	CPU	CPU	Gap MG(%)	CPU	Gap MG (%)	Gap MC (%)	Gap MT (%)
Z1	25	1	1.7	0.000	11.0	6.6	**0.0**	**-3.4**	**-7.7**
		2	0.8	0.012	16.1	6.9	**0.0**	**-8.0**	**-6.3**
	50	1	13.4	0.015	22.6	25.2	0.2	**-2.1**	**-5.0**
		2	18.8	0.031	19.4	24.4	0.3	**-1.3**	**-4.1**
	75	1	131.4	0.047	26.3	53.5	0.8	0.3	**-4.3**
		2	73.4	0.047	27.9	57.0	1.7	1.0	**-4.3**
Z2	25	1	1.2	0.015	22.6	5.5	**0.0**	**-6.6**	**-10.0**
		2	1.1	0.003	22.8	6.0	**0.0**	**-1.7**	**-8.7**
	50	1	22.7	0.016	33.6	26.6	**0.0**	**0.0**	**-9.4**
		2	13.3	0.009	20.3	25.0	0.4	**-1.9**	**-7.9**
	75	1	174.1	0.015	36.5	52.5	1.4	**-9.2**	**-2.7**
		2	102.6	0.016	36.2	48.2	1.4	0.4	**-3.1**

Table 5.5 reports results obtained for LL instances. It shows that using the adapted savings algorithm, the average computing time is 0.003 seconds for instances with 25 customers and 100 nodes, 0.008 seconds for instances with 50 customers and 100 nodes and 0.014 seconds for instances with 50 customers and 200 nodes. The average gaps are respectively 16.7%, 24.9% and 87.3%.

Table 5.6 reports results for instances based on real instances. It is clear that a feasible solution is computed in at most 0.05 seconds. The average gap between the obtained solutions and optimal ones is almost 25%.

5.5.3.2 Evaluation of the ALNS heuristic

As shown in Tables 5.2 and 5.3, 50 out of 66 optimal solutions are obtained using the ALNS heuristic for adapted Solomon [126] instances. All optimal solutions for instances with 25 customers are obtained within 10 *seconds* in average. The ALNS procedure reduces significantly the "gap MG" values compared to those associated with the adapted savings algorithm. Solutions for instances with 50 customers are in average 0.42% far from the optimal ones (33.5% using the adapted savings algorithm). The average computing time for the ALNS procedure is 35 *seconds*.

Compared to the branch-and-price algorithm for the multigraph based VRPTW, the ALNS heuristic provides near optimal solutions for all adapated Solomon [126] instances in reasonable computing times. Good solution for 24 instances, for which the exact method fails in finding a feasible solution within the time limit (7, 500 *seconds*), are obtained in 50.7 *seconds* in average. The average computing time using the heuristic is 49 times better than using the exact method for instances with 25 customers and is 31 times better for instances with 50 customers.

Results for Letchford et al. [89] are summarized in Table 5.4. The ALNS heuristic pro-

vides optimal solution for 23 out of 27 instances for which the exact method found a solution. The average "Gap MG" is 0.2% and in the worst case it reaches 1.8%. The average computing time with the ALNS heuristic is 12 *seconds* while it reaches 462 *seconds* for optimal solutions. All instances are solved within 32 *seconds* in the worst case using the ALNS while no feasible solution was found within the time limit using the branch-and-Price algorithm for 3 instances.

From Table 5.5, it comes out that using the ALNS all LL instances with 25 customers and 13 (out of 30) LL instances with 50 customers are solved to optimality. Using the ALNS procedure, the average "Gap MG" is reduced from 26.9% (for the initial solution provided by the adapted savings algorithm) to 0.1%. In the worst case, the "Gap MG" reaches 1.0%. Compared to the exact method for the multigraph representation, the ALNS heuristic is 44 times faster in average. The computing time is almost 15 *seconds* in average and reaches 41.3 *seconds* while it is almost 640 *seconds* in average and reaches 6776 *seconds* for the exact method.

Table 5.6 shows that using the ALNS heuristic all real instances with 25 customers are solved to optimality, solutions for instances with 50 and 75 customers are, respectively, 0.1% and 1.3% worse than optimal ones in average. As observed for other set of instances, average computing times are reasonable and increase with the number of customers; the average computing times for instances with 25, 50 and 75 customers are respectively 6.2 , 24.8 and 52.4 *seconds* (1.2, 17.1 and 120.4 *seconds* with the branch-and-price algorithm).

5.5.3.3 Impact of the multigraph representation

Using the multigraph representation, the ALNS heuristic improves the solution cost for 33 out of 72 adapted Solomon [126] instances on min-cost graphs (solved within time limit) and for 68 out of 74 instances on min-time graphs (Tables 5.2 and 5.3). Costs are reduced up to almost 13% and 59% against costs on graphs with, respectively, less costly paths and fastest paths. By using the ALNS heuristic, the solution cost is reduced for 6 instances on min-cost graphs and for 10 instances on min-time graphs for which the branch-and-price with the multigraph fails to compute a feasible solution in $7,500$ *seconds*.

From Table 5.4, we observe that the ALNS heuristic with the multigraph representation for Letchford et al. [89] instances improves the cost for 5 out of 21 optimal solutions on min-cost graph and for 25 out of 29 optimal solutions on min-time graph. This improvement reaches 3.7% for min-cost graph and 11.8% for min-time graph. Obtained solutions with the ALNS are in average 0.5% and 3.7% better than optimal ones with less costly paths and fastest paths.

Using the heuristic with the multigraph representation, solutions for 26 LL instances on

min-cost graph and 41 LL instances on min-time graph are improved (see Table 5.5). The cost savings are 2.1% and 3.9% in average and reach up to 8.8% and 13.5% compared to optimal solutions with least costly paths and fastest paths.

For real instances, alternative paths improve the solution costs for 9 and 12 instances compared to solutions with less costly paths and with fastest paths. This improvement reaches 9.2% for min-cost graph and 10% for min-time graph. The average cost savings are 4.9% and 8.2% for instances with 25 customers, 1.4% and 6.7% for instances with 50 customers and 1.9% and 3.6% for instances with 75 customers.

5.5.4 Sensitivity analysis

In this section, we present a sensitivity analysis to investigate the impact of the proposed insertion and removal operators on the efficiency of the search procedure. In particular, we examinate the effect of the deviation term introduced in the similarity measure for the Shaw removal heuristic (see Section 5.4.3.1) and we analyze the impact of the non-myopic insertion heuristic on the solution method. For this, we perform a set of experiments using adapted Solomon [126] instances and LL instances with 100 nodes in the original network.

First, we focus on the effect of the deviation term on the performance of Shaw removal heuristic. For this, we performed two sets of experiments using the Shaw removal heuristic and the greedy insertion heuristic: in the first set, the deviation term is ignored when evaluating similarities (β set to 0) and in the second set, the deviation term is considered ($\beta \neq 0$). In both sets, we used parameters values presented in Table 5.1. Since we use only one removal heuristic and one insertion heuristic our method is reduced to a Large Neighborhood Search (LNS). For each instance and each setting, the LNS is applied five times.

Tables 5.7 and 5.8 report respectively results for adapted Solomon [126] instances and LL instances. Column "Gap" reports the average gap between solutions obtained using the corresponding LNS procedure and best known solutions. Column "CPU" reports average computing times in seconds. Best and average values for all outputs depending on the used settings are represented, respectively, in columns "Min" and "Avg".

From Tables 5.7 and 5.8 we observe that by introducing the deviation term ($\beta \neq 0$), the solution cost is improved for all instances. The reduction on gap is up to 1.7% for best values (Table 5.8, strongly correlated instances with 50 customers) and is up to 1.6% for average values (Table 5.7, non-correlated instances with 50 customers). This can be explained by the fact that considering the deviation term ($\beta \neq 0$) tightens the relatedness measure, thus, improves the performance of the removal strategy. However, considering the deviation term increases the computing times: the average computing time for adapted Solomon [126] instances with 50 customers and NC is 51.7 seconds for the case $\beta = 0$ and is 52.4 seconds

Table 5.7: *Impact of deviation term on Shaw Removal performance with adapted Solomon [126] instances*

		Min				Avg			
		$\beta = 0$		$\beta \neq 0$		$\beta = 0$		$\beta \neq 0$	
# cust	Corr	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU
25	NC	0.6 %	10.6	0.5 %	10.6	1.3 %	11.1	0.9 %	11.1
	WC	0.3 %	4.9	0.2 %	4.9	0.9 %	5.1	0.6 %	5.1
	SC	0.2 %	2.9	0.1 %	2.9	0.8 %	3.0	0.5 %	3.0
50	NC	6.1 %	49.9	4.5 %	50.2	8.2 %	51.7	6.6 %	52.4
	WC	3.5 %	19.7	3.2 %	19.8	4.8 %	21.4	4.1 %	21.9
	SC	1.7 %	9.3	1.2 %	9.3	2.6 %	9.6	2.2 %	9.7

Table 5.8: *Impact of deviation term on Shaw Removal performance with LL instances*

		Min				Avg			
		$\beta = 0$		$\beta \neq 0$		$\beta = 0$		$\beta \neq 0$	
# cust	Corr	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU
25	NC	0.5 %	5.4	0.3 %	5.3	0.5 %	5.5	0.3 %	5.5
	WC	0.0 %	4.0	0.0 %	4.0	1.0 %	4.3	0.8 %	4.3
	SC	0.6 %	3.1	0.6 %	3.1	0.8 %	3.2	0.7 %	3.3
50	NC	4.7 %	16.9	4.1 %	16.8	5.2 %	17.7	4.2 %	17.4
	WC	2.0 %	11.2	1.4 %	11.2	4.4 %	11.8	3.4 %	12.0
	SC	6.3 %	9.2	4.6 %	9.5	7.1 %	9.9	6.3 %	10.1

for the case $\beta \neq 0$. Recall that to compute the deviation term for a pair of customers we have to enumerate possible insertion positions for each customer which justify the increase in computing time. Though, this increase is not so significant. From Tables 5.7 and 5.8, we note that for almost all instances with 25 customers computing times are the same for both cases $\beta = 0$ and $\beta \neq 0$. The increase in computing times for instances with 50 customers is 0.7 seconds in the worst case. These observations illustrate the efficiency of the proposed data structure and procedures to evaluate elementary operations and in particular to evaluate insertion operations.

Second, we investigate the impact of the non-myopic insertion heuristic and its contribution in the search procedure. We perform two sets of experiments: in the first set we apply the complete ALNS procedure (with the non-myopic heuristic) and in the second set we do not consider the non-myopic heuristic in the ALNS procedure. In Tables 5.9 and 5.10, we compare the solution quality and computing time for the two heuristics. In column "Min", gap ("Gap") and computing time ("CPU") for the best solution obtained over five runs are reported for each solution method. In column "Avg", average gaps ("Gap") and computing times ("CPU") are reported. In Tables 5.11 and 5.12, we report the contribution of each insertion-removal combination in the search scheme of the complete method. Four different criteria are analyzed to evaluate the contribution of each combination: the number of solutions that improve the best solution (row "Best solutions"), the number of accepted solutions that improve the current solution (row "Improving solutions"), the number of accepted solutions that do not improve the current solution (row "Non-Improving solutions") and the total computing time used by each combination along the search procedure (row "Computing time"). These criteria are expressed in percentage to show the contribution of each removal-

Table 5.9: *Impact of non-Myopic insertion heuristic for adapted Solomon [126] instances*

		Min				Avg			
		With non-myopic		Without non-myopic		With non-myopic		Without non-myopic	
# cust	Corr	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU
25	NC	0.0 %	12.1	0.2 %	8.7	0.1 %	16.3	0.6 %	8.9
	WC	0.0 %	7.4	0.1 %	4.6	0.0 %	8.4	0.3 %	4.7
	SC	0.0 %	4.7	0.0 %	3.1	0.0 %	5.7	0.1 %	3.2
50	NC	0.9 %	66.6	2.7 %	34.9	2.2 %	84.6	3.8 %	35.8
	WC	0.3 %	25.6	1.7 %	17.1	1.6 %	42.8	2.6 %	17.5
	SC	0.0 %	12.9	0.5 %	8.9	0.6 %	25.5	1.2 %	9.4

Table 5.10: *Impact of non-Myopic insertion heuristic for LL instances*

		Min				Avg			
		With non-myopic		Without non-myopic		With non-myopic		Without non-myopic	
# cust	Corr	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU
25	NC	0.0 %	6.7	0.0 %	5.0	0.0 %	7.4	0.0 %	5.1
	WC	0.0 %	4.6	0.0 %	3.9	0.2 %	4.9	0.2 %	3.9
	SC	0.0 %	3.6	0.0 %	3.2	0.2 %	4.2	0.5 %	3.3
50	NC	0.1 %	22.6	5.0 %	15.1	1.0 %	24.2	5.0 %	15.4
	WC	0.1 %	14.2	2.6 %	11.2	0.4 %	15.6	4.2 %	11.5
	SC	0.3 %	11.1	3.8 %	9.7	1.2 %	13.2	5.3 %	10.0

insertion combination.

From Tables 5.9 and 5.10 it comes out that the non-myopic heuristic improves significantly the solution quality. The gap between the best solution and the optimal solution is up to 5.0% for the method without the non-myopic insertion heuristic, while, it is 0.9% in the worst case with the complete ALNS procedure. The average gaps for adapted Solomon [126] instances with 25 and 50 customers are, respectively, 0.03% and 1.46% with the non-myopic heuristic and are 0.33% and 2.53% without the non-myopic heuristic. For LL instances, using the complete search procedure average gaps are 0.13% and 0.86% and are 0.23% and 4.83% using the solution method without the non-myopic insertion heuristic. However, by considering the non-myopic heuristic in the search procedure the computing time increases considerably: for adapted Solomon [126] instances with 50 customers and no correlation, a solution is computed in average in 84.6 seconds using the complete ALNS procedure and in 35.1 seconds without the non-myopic insertion heuristic. The average increases in computing time are up to 4.5 and 30.1 seconds for adapted Solomon [126] instances with, respectively, 25 and 50 customers and are 1.4 and 5.4 seconds for LL instances with, respectively, 25 and 50 customers.

Table 5.11: *Contribution of Removal-insertion combinations on the search scheme for adapted Solomon [126] instances*

Removal heuristic		Random				Worst				Shaw			
Insertion heuristic		Greedy	Regret	Non-myopic	Simple	Greedy	Regret	Non-myopic	Simple	Greedy	Regret	Non-myopic	Simple
25	Best solutions	5.1 %	10.7 %	8.1 %	0.0 %	10.3 %	22.6 %	15.5 %	0.0 %	4.8 %	13.7 %	9.2 %	0.0 %
	Improving solutions	7.2 %	11.8 %	10.4 %	0.0 %	10.1 %	15.3 %	13.0 %	0.0 %	7.8 %	12.9 %	11.5 %	0.0 %
	Non-Improving solutions	10.6 %	8.8 %	9.5 %	0.1 %	14.0 %	13.1 %	13.4 %	0.2 %	11.2 %	9.0 %	9.9 %	0.1 %
	Computing time	5.5 %	6.4 %	16.8 %	2.1 %	6.2 %	7.1 %	19.0 %	2.3 %	6.5 %	7.3 %	18.3 %	2.5 %
50	Best solutions	6.1 %	8.7 %	8.0 %	0.0 %	11.9 %	21.2 %	16.1 %	0.0 %	7.3 %	11.5 %	9.2 %	0.0 %
	Improving solutions	8.0 %	10.7 %	9.6 %	0.0 %	11.4 %	15.2 %	13.3 %	0.0 %	8.9 %	12.1 %	10.8 %	0.0 %
	Non-Improving solutions	10.4 %	8.6 %	8.8 %	0.1 %	15.2 %	13.5 %	13.4 %	0.2 %	11.1 %	9.0 %	9.6 %	0.1 %
	Computing time	6.4 %	7.0 %	14.3 %	2.9 %	7.3 %	8.2 %	16.6 %	3.3 %	7.3 %	8.0 %	15.5 %	3.1 %

Table 5.12: *Contribution of Removal-insertion combinations on the search scheme for LL instances*

Removal heuristic		Random				Worst				Shaw			
Insertion heuristic		Greedy	Regret	Non-myopic	Simple	Greedy	Regret	Non-myopic	Simple	Greedy	Regret	Non-myopic	Simple
25	Best solutions	4.8 %	8.4 %	4.7 %	0.0 %	13.3 %	22.0 %	18.3 %	0.0 %	9.0 %	6.3 %	6.5 %	0.0 %
	Improving solutions	8.4 %	9.4 %	10.0 %	0.0 %	11.6 %	15.3 %	15.0 %	0.0 %	9.7 %	9.6 %	11.0 %	0.0 %
	Non-Improving solutions	10.1 %	9.9 %	8.8 %	0.1 %	14.6 %	14.6 %	13.9 %	0.1 %	9.1 %	9.9 %	8.8 %	0.0 %
	Computing time	6.5 %	7.1 %	13.6 %	3.2 %	7.5 %	8.3 %	16.5 %	3.8 %	7.3 %	7.8 %	14.9 %	3.5 %
50	Best solutions	7.2 %	5.3 %	9.6 %	0.0 %	13.5 %	16.8 %	24.1 %	0.0 %	6.7 %	3.1 %	13.7 %	0.0 %
	Improving solutions	8.1 %	6.8 %	11.6 %	0.0 %	13.1 %	15.0 %	18.1 %	0.0 %	8.5 %	5.9 %	12.9 %	0.0 %
	Non-Improving solutions	9.0 %	9.7 %	8.7 %	0.1 %	14.2 %	14.7 %	14.7 %	0.1 %	9.4 %	11.0 %	8.3 %	0.1 %
	Computing time	5.4 %	5.5 %	16.8 %	2.5 %	6.5 %	6.9 %	19.4 %	2.9 %	6.4 %	6.7 %	17.9 %	3.0 %

Tables 5.11 and 5.12 report the contribution of each combination insertion-removal heuristics to the efficiency of the complete ALNS procedure. A first observation is that for each class of instances the search procedure presents a different behaviour which confirms the adaptiveness of the search scheme. Tables 5.11 and 5.12 show that for almost all classes of instances, combinations involving non-myopic insertion heuristic have important contributions for all criteria, e.g., the combination of the worst removal and non-myopic insertion heuristics presents the best contributions over all criteria for LL instances with 50 customers: 24.1% of solutions improving the best solution, 18.1% of solutions improving the current solution, and 14.7% of visited solutions but non-improving current solution are obtained using this combination. A second observation is that combinations involving non-myopic heuristic are the most time consuming. The average computing time used by the combination of the worst removal and non-myopic insertion heuristics is up to 19.4% for LL instances with 50 customers. This explains the increase in time when considering the non-myopic insertion heuristic in the ALNS procedure (Tables 5.9 and 5.10).

5.6 Conclusion

Due to their numerous applications, Vehicle Routing Problems have drawn many researchers interest. In some real-life applications, different attributes have to be considered when defining transportation plans such as operational cost, traveling time, energy consumption, etc. In such cases, alternative links proposing different compromises may exist between each pair of points of interest on the road network. Using the customer-based graph where only one path is considered between each pair of customer locations, the decision maker may face infeasibility problems or overestimate the optimal solution cost. For these reasons, a multigraph representation would be more appropriate to accurately model the problem.

In this study, we were interested in the Vehicle Routing Problem with Time Windows on road networks, where several attributes are defined for each route segment. We proposed to adopt the idea of alternative paths and model the problem using a multigraph representation. We provided a heuristic solution method based on an Adaptive Large Neighborhood Search. The presence of parallel arcs introduces computational challenges especially in exploring the neighborhood of a given solution; elementary operations like removing or inserting a customer in a route induce an NP-hard problem, called Fixed Sequence Arc Selection Problem. To handle this issue, we integrated in our heuristic an incremental data structure and we proposed a procedure based on dynamic programming approach to evaluate neighbor solutions. We conducted an intensive experimental study on several set of instances with different characteristics. Numerical results showed the efficiency of the proposed heuristic compared to an exact method and illustrated the interesting impact of alternative paths on the solution quality. Experiments on instances generated based on data for real road network demonstrated the interest of the modeling approach and the important cost savings generated compared to a standard VRPTW solution.

The results obtained here for VRPTW confirmed and completed results reported by Garaix et al. [61], Ben Ticha et al. [8] and Lai et al. [85] for other routing problems where several attributes on road network segments. The solution method proposed here to address multigraph based VRPTW with two attributes (travel cost and time) can be easily extended to handle the more general case of multiple arc attributes. However, numerical results showed that multigraph representation could increase significantly computing time. A future study could investigate the interest of tackling these categories of problems directly in the road network. Letchford et al. [89] proposed a column generation scheme but a complete solution method still missing.

Chapter 6

A branch-and-price Algorithm for the Vehicle Routing Problem with Time Windows on a road-network graph

This chapter is a working paper.

Abstract

Vehicle routing problems are, typically, defined on road networks where customer locations are associated with a subset of nodes and arcs represent road segments. Most approaches in the literature assume that the best path between every two points in the road network can be easily defined. Thus, the problem can be tackled using a so-called customer-based graph representation, where nodes represent customers and depot locations and an arc represents the best path between two customer nodes. However, when several attributes are defined on road segments this representation can have negative effects on the solution quality. To handle these limits, two approaches have been proposed in the literature. In the first approach, the road network is represented using a multigraph where an arc is introduced for every alternative path. The second approach consists in solving the problem directly on the original road network. In this paper, we propose to investigate in depth the road network based approach. We consider the Vehicle Routing Problem with Time Windows as a test-bed problem and we develop a complete branch-and-price scheme that can handle the road network setting. An extensive computational study based on several types of instances is conducted in order to evaluate the relative efficiency of the multigraph-based approach and solving the problem directly on the original road network.

Keywords: *Vehicle Routing Problems, Road networks, Branch-and-Price.*

6.1 Introduction

The Vehicle Routing Problem (VRP) can be described as the problem of designing a set of optimal routes to be used by a fleet of vehicles to visit a set of geographically dispersed

customers. These routes must start and end at a depot location and must satisfy a set of constraints (vehicle capacity, customers' time windows, route duration, etc.). Since its introduction by Dantzig and Ramser [31], hundreds of papers and books have been devoted to study the VRP [87]. Many variants of vehicle routing problems have been proposed to address several issues that arise in real life applications such as the Capacitated Vehicle Routing Problem (CVRP) where a certain demand has to be delivered to every customer and the total load to be delivered along a route must satisfy the vehicle capacity [129], the Vehicle Routing Problem with Time Windows (VRPTW) where transportation plans are constrained to satisfy customer requests within their time windows [126], the Multi-Depot Vehicle Routing Problem (MDVRP) where vehicles are based on several depot locations [26], etc. Extensive reviews on the most common variants of the VRP are available in [65] and [128].

Conventionally, vehicle routing problems are tackled using a simple graph representation of the original road network where a node is introduced for each point of interest and an arc represents the shortest path between the endpoints. This representation, called customer-based graph, stems from the assumption that the best path between each pair of nodes in the road network can be easily defined. Yet, in practice several attributes can be defined on road segments (e.g., travel time, travel cost, energy consumption, etc.). Thus, alternative paths presenting different trade-offs could exist between each pair of points of interest. Not considering these alternatives may discard potentially good solutions from the solution space and could have a negative effect on the solution quality.

In the literature, an increasing number of papers investigate the effects of the simple graph representation on the solution quality for vehicle routing problems with several attributes are defined on road segments. Two modelling approaches have been proposed to handle these effects. The first approach consists in representing the original road network with a multigraph where nodes represent points of interest and an arc is introduced for every non-dominated path between two points of interest. The second approach consists in tackling the problem directly on a graph that mimics the road network.

To the best of our knowledge, Garaix et al. [61] were the first to point out that transforming the vehicle routing problem on a road network into a standard vehicle routing problem may lead to losing optimality when several attributes are defined on road segments. To handle this issue, they propose to consider all alternative routes using a multigraph structure and develop a branch-and-price procedure to solve a dial-a-ride problem.

More recently, Letchford et al. [89] revisited the branch-and-price approach presented in [61]. They suggest that it could be more 'natural' and more efficient to tackle the problem directly on the road network, rather than using a multigraph representation. They explain how it would impact both the pricing problem and branching rules, but only explore the pricing problem. In their computational experiments, they compare the computing times at the root node of the branch-and-price tree with the road network and the multigraph approaches.

Obtained results confirm their suggestions and illustrate the efficiency of the pricing routines on the road network compared to those with the multigraph representation.

Although the results obtained by Letchford et al. [89] are interesting, their conclusions can be hardly generalized for many reasons. First, they are interested in the multiple Travelling Salesman Problem with Time Windows (m-TSPTW). In this problem, a time window is associated with each customer and no restriction on the total load carried along a route is considered. Yet, in practice the used vehicles have a limited capacity and a demand (to be delivered or to be picked up) has to be served for every customer. This issue could have a significant impact on algorithm efficiency with both road network and multigraph approaches. Second, their experiments are based on instances with relatively high densities of customers: two sets of instances are considered with densities equal to 33% and 66% respectively. Real life applications are defined on large scale road networks in which a few number of nodes are associated with customer locations. Third, they only investigate the pricing problem where non-elementary routes are allowed. The case with only elementary routes is not explored. In addition, lower bounds at the root node of the branch-and-price tree with the road network and multigraph approaches are not guaranteed to be the same when non-elementary routes are allowed. This issue is not examined in their computational experiments and they only focus on computing times obtained with both algorithms. Finally, Letchford et al. [89] only explore the pricing problem and do not devise suitable branching rules. It is worth mentioning that the standard branching rules for vehicle routing problems can be easily adapted to handle the multigraph setting (see [61] and [8]), however, it is not the case for the algorithm that works directly on the road network. Suitable branching rules may result on a different branch-and-price scheme on the road network than with the multigraph. Thus, conclusions based on results obtained at the root node cannot be generalized for complete branch-and-price scheme. For all these reasons, further analyses and extensive comparisons on the efficiency of the branch-and-price algorithms with the road network and the multigraph approaches are needed to achieve comprehensive conclusions.

In this paper, we propose to investigate more in depth the relative efficiency of the branch-and-price algorithms with the road network and the multigraph approaches and to further analyse results reported by Letchford et al. [89]. We are interested in what is probably the simplest and the most studied vehicle routing problem with two attributes: the VRPTW. We develop a complete branch-and-price algorithm based on pricing routines presented in [89]. We base our experiments on three type of instances: (1) instances generated by Letchford et al.[89]; (2) a large set of realistic instances constructed following Letchford et al. [89]; (3) instances derived from real road networks. An extensive computational study is proposed in order to analyse the impact of different factors (capacity constraints, customers density, etc.) on the relative efficiency of both branch-and-price algorithm.

The reminder of the paper is organized as follows. In Section 6.2, we review the relevant literature. In Sections 6.3 and 6.4, we describe the branch-and-price algorithms for the road

network and multigraph approaches, respectively. Finally, in Section 6.5, we report the results obtained and we analyse the impact of both modelling approaches on the efficiency of the algorithm.

6.2 Literature review

Vehicle routing problems have drawn many researchers' attention for more than fifty years. A large number of solution methods are proposed in the literature to solve different variants. Most of these approaches are based on the key assumption that the best origin-destination path can be computed for every pair of points of interest. Thus, the problem can be addressed using a customer-based graph: a node is introduced for each point of interest (customer or depot) and arcs represent the best paths. However, this assumption is not guaranteed to hold when several attributes are defined on road segments. In this case, the customer-based graph representation could have negative impact on the solution quality.

As mentioned before, this issue was, first, investigated by Garaix et al. [61]. They show that transforming the vehicle routing problem on a road network into a standard vehicle routing problem may result in losing optimality. They are interested in efficiently solving a Dial-a-Ride Problem. They propose a branch-and-price scheme that can handle the multigraph setting. Later, Ben Ticha et al. [8] examine more in depth the impact of the multigraph representation on the solution quality for vehicle routing problems when several attributes are defined on road segments. They revisite the branch-and-price algorithm proposed by Garaix et al. [61] and consider the VRPTW as a test-bed problem. An experimental analysis is conducted based on several types of instances: modified instances from the literature and instances derived from real road networks data. They report results that confirm the impact of the multigraph on the solution quality compared to two customer-based graph representations: a fastest-path-based graph and a cheapest-path-based graph.

In the literature, a few heuristic approaches have also been proposed to show the interesting impact of the multigraph representation. In [85] a tabu search heuristic is proposed for the Heterogeneous VRP with limited route duration. Ben Ticha et al. [7] develop an adaptive large neighborhood search procedure for the VRPTW.

Letchford et al. [89] confirm the negative effect of the traditional customer-based graph representation on the solution quality for vehicle routing problems when several attributes are defined on road segments. They suggest that it could be more interesting to tackle the problem directly on the road network instead of using a multigraph representation. They explain how it would impact a branch-and-price scheme and propose two procedures to solve the pricing problem directly on the road network: the first procedure generates only elementary routes and in the second non-elementary routes are allowed. But, they only experiment the

case where non-elementary routes are allowed. They show that for both cases the running times for the road network based algorithms compares favourably with the running times that would be obtained using the multigraph representation in the worst case. In their experiments, they demonstrate the interest of their approach.

Besides the study presented in [89], a stream of papers concerned with vehicle routing problems on road networks focus on the so-called Steiner Travelling Salesman Problem (STSP). The STSP was introduced, independently, by Orloff [103], Fleischmann [56] and Cornuéjols et al. [29]. It can be defined as the problem of finding the min-cost cycle visiting a set of required nodes in a road network. The motivation of tackling the problem directly on the original road network instead of using a complete graph representation is to exploit any property that the road network may present such as sparsity or planarity. Fleischmann [56] states that nodes in road networks have small degrees and adding artificial arcs to complete the original graph increases dramatically the number of variables needed in the linear programming formulation. He proposes a cutting-plane approach to solve the problem and suggests a way to extend the solution procedure to the VRP but reveals some difficulties that would arise in this case. Cornuéjols et al. [29] formulate the problem as an integer program with a linear number of variables and an exponential number of constraints then solve the problem using a branch-and-bound algorithm. Recently, compact formulations for the STSP with linear numbers of variables and constraints have been proposed by Letchford et al. [90].

It is worth mentioning that the most part of the literature of vehicle routing problems on road network concerns arc routing problems where transportation requests are associated with arcs [44, 24]. In this context, a relevant study is proposed by Letchford and Oukil [91]. The objective of this study is to show that, by exploiting its sparsity, it is more efficient to tackle the routing problem directly on the road network instead of using a complete graph representation. The authors are interested in the Capacitated Arc Routing Problem (CARP) and investigate the pricing problem in a branch-and-price scheme. Two dynamic programming algorithms are proposed for the problem with only elementary routes and where non-elementary routes are allowed. Recently, a complete branch-and-price algorithm for the CARP on the road network was developed by Bode and Irnich [12]. They adopted the pricing routines proposed in [91] and propose an adapted branching rule for the CARP that can handle the road network setting.

In this paper, we consider the VRPTW with two attributes on road segments as a test-bed problem. The VRPTW on the road network is defined as follows. Let $G = (V, A)$ be a directed road network where V is the set of n nodes and A is the set of arcs. With each arc $(i, j) \in A$ is associated a travel time t_{ij} and a travel cost c_{ij}. Let node 0 represents the depot location and $C \subset V \setminus \{0\}$ represents the set of customers. With each customer $i \in C$ is associated a demand d_i, a time window $[e_i, l_i]$ and a service time s_i. A homogeneous fleet of K vehicles with a loading capacity Q is given. The aim is to find a set of routes of minimal total cost, starting and ending at the depot, and that serves each customer exactly once.

The standard VRPTW has been intensively studied in the literature. A taxonomic review of the vehicle routing literature published between 2009 and 2015 shows that up to 38% of articles consider time windows as a physical characteristic of customers [13]. Numerous exact approaches are proposed to solve the standard VRPTW [25, 2]. These approaches include Lagrangian relaxation [79], branch-and-cut [94, 4], branch-and-price [40]. Among these approaches, the branch-and-price scheme has been widely investigated [83, 18, 77, 38, 59].

This literature revue confirms that there is a lack of papers that address vehicle routing problems on road networks. Except the study presented by Letchford et al. [89], there is no study that investigates the impact of tackling vehicle routing problems directly on the road network on the solution quality and how it would impact the performance of solution methods. This paper makes a number of contributions to the literature. First, it proposes a complete branch-and-price scheme that solves the VRPTW directly on the road network. As we mentionned before, Letchford et al. [89] explore only the pricing problem. Branching rules that work on the original graph are needed. Although Bode and Irnich [12] develop such rules for the CARP, we show in Section 6.3.2 that the proposed scheme is not suitable for the VRPTW with several attributes on road segments. Second, it completes the results presented by Letchford et al. [89] and investigates more in depth the solution of vehicle routing problems on the road network. An extensive computational study is conducted in order to evaluate the impact of tackling the problem directly on the road network instead of using a multigraph. We base our experiments on two sets of instances. The first set of realistic instances are generated as in [89] and with several densities of customers and the second set consists of instances derived from real-world road networks.

6.3 Branch-and-price algorithm for the VRPTW on the road-network graph

As for the standard VRPTW, the branch-and-price scheme for the VRPTW on the road network is based on the following set covering formulation:

$$Min \sum_{r \in \Omega} c_r x_r \tag{6.1}$$

$$s.t \quad \sum_{r \in \Omega} a_{i,r} x_r \geq 1 \quad \forall i \in C \tag{6.2}$$

$$\sum_{r \in \Omega} x_r \leq K \tag{6.3}$$

$$x_r \in \{0, 1\} \quad \forall r \in \Omega \tag{6.4}$$

where Ω represents the set of feasible routes in the road network, c_r represents the cost of route $r \in \Omega$ and a_{ir} is a binary parameter equals to 1 if and only if the customer $i \in C$ is served along the route r. The binary decision variable x_r takes the value 1 if the route r is selected in the optimal solution and 0 otherwise.

Due to the exponentially growing size of Ω, the optimal solution for the LP relaxation of (6.1–6.4), so-called Master Problem (MP), cannot be computed using a standard branch-and-bound procedure. This issue is handled using a column generation technique embedded into the branch-and-bound framework.

In the column generation procedure, only a subset of columns $\Omega_1 \subset \Omega$ is considered and a restriction of the Master Problem MP to $\Omega_1 \subset \Omega$, denoted by $MP(\Omega_1)$, is solved at each iteration. Ω_1 consists of all the columns generated at previous iterations and is iteratively enlarged by solving the pricing problem. The pricing problem aims at finding new routes with negative reduced costs, *i.e.*, routes that offer better ways to visit customers. For a detailed description of the column generation algorithm, the reader can refer to [52].

6.3.1 Pricing problem

Let λ_i, $i \in C$, be the dual variable associated with constraints (6.2) and λ_0 be the dual variable associated with constraint (6.3). The reduced cost of a route r is given by:

$$\hat{c_r} = c_r - \sum_{i \in C} a_{i,r}\lambda_i - \lambda_0 \tag{6.5}$$

The purpose of the pricing problem is to generate routes $r \in \Omega \setminus \Omega_1$ with $\hat{c_r} < 0$. Note that in the standard version of the VRPTW, the pricing problem can be easily reduced to an Elementary Shortest Path Problem with Resource Constraints (ESPPRC). This is done by replacing c_r and $a_{i,r}$ in (6.5), respectively by $\sum_{(i,j)\in A} b_{ijr}c_{ij}$ and $\sum_{j\in V|(i,j)\in A} b_{ijr}$ (recall that $V = C \cup \{0\}$ in the standard VRPTW) with $b_{ijr} = 1$ if route r traverses arc (i, j) and 0 otherwise. Using these notations, the reduced cost of route r can be expressed as:

$$\hat{c_r} = \sum_{(i,j)\in A} b_{ijr}(c_{ij} - \lambda_i) \tag{6.6}$$

Thus, the pricing problem is equivalent to the problem of finding elementary paths starting and ending at the depot 0, satisfying time windows and capacity constraints and with negative cost where arc costs are equal to $c_{ij} - \lambda_i$. This problem can be efficiently handled using a dynamic programming based approach [53, 115]. Desrochers et al. [40] show that the ESPPRC is NP-hard in the strong sense, however it can be solved in a pseudo-polynomial time when the elementary condition is not considered and in this case the pricing problem is reduced to a Shortest Path Problem with Resource Constraints (SPPRC). They notice that,

with the SPPRC, the enlargement of the set of feasible routes Ω does not affect the validity of the set covering formulation and the branch-and-price scheme, but, it weaken significantly the lower bound. Also, slight modifications must be made to the branch-and-price framework: b_{ijr} represents the number of times the arc (i, j) is traversed along the route r and $a_{i,r}$ represents the number of times the customer i is served in r.

With the road network setting, two main issues arise. First, a customer node can be visited more than once in a route (but it is not necessarily served at each visit). Second, an arc can be traversed several times along a route. Thus, the pricing problem cannot be reduced to an ESPPRC. In this case, a route r is called elementary if every customer in r is served only once otherwise it is called non-elementary. To handle the road network setting, the branch-and-price framework is modified as follows: b_{ijr} represents the number of times the arc (i, j) is traversed along the route r and $a_{i,r} = 1$ if the customer i is served in r otherwise it is equal to 0.

To solve the pricing problem, we adapt the algorithm proposed by Feillet et al. [53]. The solution procedure is based on a modified label correcting algorithm. A label represents a partial route from the depot node 0 to a node $v \in V$. It is defined using the following information, $L = (v, t, c, q, S)$ with:

- v is the last node visited in the partial route represented by L;
- t represents the arrival time at v. When v is a served customer, t includes waiting and service times;
- c represents the reduced cost of the partial route represented by L;
- q is the total demand of served customers along the partial route represented by L;
- S represents the set of served customers along the partial route represented by L. S contains also the unreachable customers, *i.e.* that cannot be served along the partial route represented by L without violating time or capacity constraints.

The developed algorithm is based on an exhausitive enumeration in which, for every label, all feasible extensions are performed. Once all labels are processed the algorithm terminates and all routes with negative reduced costs are constructed. Using this search strategy, the number of generated labels can be very large. To reduce this number, a dominance check is used. The dominance rule is defined as follows:

Definition 6.1. *A label $L_1 = (v, t_1, c_1, q_1, S_1)$ dominates a label $L_2 = (v, t_2, c_2, q_2, S_2)$ if:*

1. $t_1 \leq t_2$
2. $c_1 \leq c_2$
3. $q_1 \leq q_2$

4. $S_1 \subseteq S_2$

The general scheme of the pricing algorithm is described in **Algorithm 6.1** where $Labels[v]$ is the list of labels whose last visited node is v and W is the list of active nodes *i.e.* at which there is a non-processed label. $Labels[v].insert(L)$ is a procedure that updates labels in $Labels[v]$ and keeps only non-dominated labels. This procedure returns '*False*' if L is dominated and has not been inserted. $W.extract()$ is a procedure that extracts a node from W. $L.updateUnreachableNodes$ is a procedure that updates the set of unreachable nodes along the partial route represented by L. In **Algorithm 6.1**, when the destination node is a customer $v \in C$, every label is extended twice: in the first extension the customer v is served, if it is possible regarding time and capacity constraints, and in the second extension v is visited without being served.

Note that, the pricing procedure for the road network based ESPPRC can be easily adapted to solve the SPPRC. To do this, we no longer need to check if customer v was already served at each extension of a label along an arc $(u, v) \in A$ (line 11 of **Algorithm 6.1**) and the comparison based on the set of served customers is not considered when checking the dominance between two labels.

6.3.2 Branching scheme

The branching rule is one of the important components of the branch-and-price scheme. It aims at adding constraints in order to iteratively extend the binary search. It is important to ensure that the added constraints are compatible with the column generation procedure.

In the context of vehicle routing problems, the standard branching rule relies on the property that in a feasible solution each arc is at most traversed by one vehicle (see [74] for more details). Let ϕ_{ij} denotes the flow on arc (i, j), *i.e.*, $\phi_{ij} = \sum_{r \in \Omega_1} b_{ijr} x_{ij}$. The standard branching rule consists in selecting an arc (i, j) such that $0 < \phi_{ij} < 1$ then, deriving two branches: in the first branch, the use of the arc (i, j) in the solution is forbidden, and in the second branch, the arc (i, j) is enforced in the solution. This branching rule is very easy to address in both column generation schemes and in the Master Problem.

Unfortunately, the standard branching rule cannot be used with the road network, due to the fact that an arc can be traversed several times and by several vehicles. In our implementation, we propose to use a branching rule that works as follows:

- Select an arc $(i, j) \in A$ with fractional flow $\phi_{ij} > 0$
- Derive two branches:
  - In the first branch, the upper limit on flow on arc (i, j) is fixed to $\lfloor \phi_{ij} \rfloor$
  - In the second branch, the lower limit on flow on arc (i, j) is fixed to $\lfloor \phi_{ij} \rfloor + 1$

Algorithm 6.1 Pricing procedure for based road network ESPPRC

1: $L = (0, 0, 0, 0, \emptyset)$
2: $Labels[0].add(L)$
3: **for all** $i \in V$ **do**
4: $\quad Labels[i] \leftarrow \emptyset$
5: **end for**
6: $W = \{0\}$
7: **while** $W \neq \emptyset$ **do**
8: $\quad u = W.extract()$
9: $\quad$ **for all** $L = (u, t, c, q, S) \in Labels[u]$ **do**
10: $\quad\quad$ **if** L is not processed **then**
11: $\quad\quad\quad$ **for all** $(u, v) \in A$ **do**
12: $\quad\quad\quad\quad$ **if** $v \in C \cup \{0\}$ and $v \notin S$ **then**
13: $\quad\quad\quad\quad\quad$ **if** $t + t_{uv} \leq l_v$ and $q + q_v \leq Q$ **then**
14: $\quad\quad\quad\quad\quad\quad L' = (v, min\{e_v, t + t_{uv}\} + s_v, c + c_{uv} - \lambda_v, q + q_v, S' = S \cup \{v\})$
15: $\quad\quad\quad\quad\quad\quad L'.updateUnreachableNodes()$
16: $\quad\quad\quad\quad\quad\quad$ **if** $Labels[v].insert(L')$ **then**
17: $\quad\quad\quad\quad\quad\quad\quad W \leftarrow W \cup \{v\}$
18: $\quad\quad\quad\quad\quad\quad$ **end if**
19: $\quad\quad\quad\quad\quad$ **end if**
20: $\quad\quad\quad\quad$ **end if**
21: $\quad\quad\quad\quad L'' = (v, t + t_{uv}, c + c_{uv}, q, S)$
22: $\quad\quad\quad\quad L''.updateUnreachableNodes()$
23: $\quad\quad\quad\quad$ **if** $Labels[v].insert(L'')$ **then**
24: $\quad\quad\quad\quad\quad W \leftarrow W \cup \{v\}$
25: $\quad\quad\quad\quad$ **end if**
26: $\quad\quad\quad$ **end for**
27: $\quad\quad$ **end if**
28: $\quad$ **end for**
29: **end while**
30: **return** $Labels[0]$

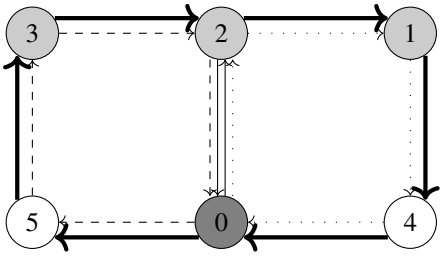


Figure 6.1: *Example of a fractional solution supported by an integer arc flow*

This branching rule is very easy to address in the column generation scheme. Constraints

$$\sum_{r\in\Omega_1} b_{ijr}x_r \leq \lfloor \phi_{ij} \rfloor \tag{6.7}$$

and

$$\sum_{r\in\Omega_1} b_{ijr}x_r \geq \lfloor \phi_{ij} \rfloor + 1 \tag{6.8}$$

are added to the Master Problem in the first and second branches, respectively. At the pricing problem level, the cost on arc (i, j) is set to $c_{ij} = c_{ij} - \lambda_{ij}^{low}$ (resp. $c_{ij} = c_{ij} - \lambda_{ij}^{up}$) in the first branch (resp. the second branch) with $\lambda_{ij}^{low} \leq 0$ is the dual variable associated with constraint (6.7) (resp. $\lambda_{ij}^{up} \geq 0$ is the dual variable associated with constraint (6.8)).

Note that, it has been shown in [5] that for the standard VRPTW, an integer flow corresponds to an integer routing solution. Thus, when no arc with a fractional flow is found, the obtained solution corresponds to the optimal solution of the associated branch. With the road network setting, this property does not hold. A fractional routing solution could be supported by an integer flow. To illustrate this, let us consider the example provided in **Figure 6.1**.

In this example, the depot is located at node 0, and customers are located at nodes 1, 2 and 3. Other nodes represent roads junctions. Suppose that we are given a fleet with a large number of vehicles and let us consider the following routes (assumed to be feasible regarding time windows and capacity constraints):

- r_1 (represented with thin lines) starts from the depot serves the customer at node 2 and ends at the depot;
- r_2 (represented with dashed lines) serves the customer at node 3;
- r_3 (represented with dotted lines) serves the customer at node 1;
- r_4 (represented with thick lines) serves, successively, customers at nodes 3, 2 and 1.

The solution defined by $x_{r_1} = 0.5$, $x_{r_2} = 0.5$, $x_{r_3} = 0.5$ and $x_{r_4} = 0.5$ represents a feasible solution for the Master Problem since $\sum_{r_k;k\in\{1,\ldots,4\}} a_{i,r_k}x_{r_k} \geq 1$ is satisfied for all customer nodes $i \in C$ and $\sum_{r_k;k\in\{1,\ldots,4\}} x_{r_k} = 2 \leq K$ (assumed to be large). We can see that the flow associated

with this solution is integer but it does not correspond to a feasible routing solution.

To handle such situations, we propose a procedure based on a specific branching scheme. Suppose that at a certain node of the search tree, the obtained solution $\tilde{x} = (\tilde{x}_r)_{r\in\Omega_1}$ is fractional and the associated flow is integer, *i.e.*, $\sum_{r\in\Omega_1} b_{ijr}\tilde{x}_r$ is integer for all arcs $(i, j) \in A$. And let $\tilde{G} = (V, \tilde{A})$ the graph induced by this solution, *i.e.*, $\tilde{A} = \{(i, j) \in A; \sum_{r\in\Omega_1} b_{ijr}\tilde{x}_r > 0\}$. The proposed procedure derives two branches:

- In the first branch, we seek for a feasible solution defined on the graph $\tilde{G}$. For this aim, we apply the Algorithm 6.1 (in which the dominance rule is deactivated in the insertion procedure) to enumerate the complete set of feasible routes that may exist in $\tilde{G}$, denoted by $\tilde{\Omega} \subset \Omega$. Then, a set covering problem is solved based on the set $\tilde{\Omega}$ with an IP solver;
- In the second branch, we enforce the use of at least one arc that is not used by the solution $\tilde{x}$. To do this, constraint (6.9) is added to the master problem.

$$\sum_{(i,j)\in A\setminus\tilde{A}} \sum_{r\in\Omega_1} b_{ijr}x_r \geq 1 \tag{6.9}$$

  At the pricing problem level, a dual variable $\tilde{\lambda}_a \geq 0$ is subtracted from the cost of each arc a considered in constraint (6.9).

It is worth mentioning that Bode and Irnich [12] propose a branching scheme for the CARP that can handle the road network setting. The proposed branching scheme is based on three levels of decisions: (1) branching on node degrees, (2) branching on edge flows and (3) branching on *followers and non-followers*. The first level consists in branching once a node with a non-even (fractional or odd) degree is found. The second level of branching is similar to the one proposed in this paper. These two branching rules are mainly used to improve the quality of the lower bound and do not guarantee the integrality of the solution. The third branching rule consists in deciding whether two required edges are serviced consecutively in the same route or not. To do this, an undirected *follower information* $f_{ee'}$ is defined for each pair of required edges (e, e') and is given by $f_{ee'} = \sum_{r\in\Omega} f_{ee'r}x_r$ with $f_{ee'r}$ denotes the number of times edges e and e' are served consecutively along the route $r \in \Omega$. The *followers and non-followers* branching scheme relies on the property that in a feasible solution $f_{ee'} \in \{0, 1\}$ for every pair of required edges (e, e'). Thus, if two required edges e and e' with $f_{ee'}$ are found, two branches are with constraints $f_{ee'} = 0$ and $f_{ee'} = 1$ derived. The first type of constraints consists in forbidding servicing edge e' immediately after servicing edge e. This is done by removing all routes satisfying this property from the Master problem and to avoid pricing new routes with this property the same task is associated with edges e and e' and a *task-2-loop* elimination technique (see [75], [74]) is used to ensure that the constraint $f_{ee'} = 0$ is respected. On the *follower* branch $f_{ee'} = 1$, only routes where the successor of e (resp. e') is not e' (resp. e) are not considered in the Master Problem, and at the pricing problem level, edges e and e' are replaced by four new edges that model the consecutive service to e and e'.

Bode and Irnich [12] show that the *followers and non-followers* branching rule can ensure the integrality of the routing solution for the CARP. Unfortunately, this branching rules are not very suitable for the problem considered here. This is because several alternative paths can exist between each pair of customer nodes. To address the *follower* constraint between two nodes i and j, one has to compute all non-dominated paths linking i to j which involves an NP-hard problem called multi-objective shortest path problem [121]. In addition, it has been shown that the number of Pareto-optimal paths can be exponential in the number of nodes in the network which would be very computationally expensive for the pricing problem.

6.4 Branch-and-price algorithm for the multigraph based VRPTW

In this section, we briefly describe the branch-and-price scheme for the VRPWT with the multigraph representation. A full description can be found in [8].

The multigraph-based VRPTW is defined on a directed multigraph $G^{MG} = (V^{MG}, A^{MG})$ consisting of $V^{MG} = C \cup \{0\}$ a set of nodes. The set of arcs A^{MG} contains parallel arcs between each pair of nodes: $A^{MG} = \bigcup_{i,j \in V^{MG}} A^{MG}_{ij}$ with $A^{MG}_{ij} = \{(i,j)^p, p = 1, \ldots, m_{ij}\}$ represents the set of alternative paths linking customer locations i and j in the road network. Each arc $(i,j)^p \in A^{MG}$ is given a travel time t^p_{ij} and a travel cost c^p_{ij} that represent respectively the time needed and the cost induced to travel from i to j using the path indexed by p.

With the multigraph representation, the Master problem is the same as for the standard VRPTW. However, some modifications have to be made for the column generation scheme. First, the addressed pricing problem is a multigraph based ESPPRC where the task is to generate elementary routes that satisfy:

$$\sum_{(i,j) \in A^{MG}} \sum_{p=1}^{|A_{ij}|} \alpha^r_{ijp} (c^p_{ij} - \lambda_i) \tag{6.10}$$

with $\alpha^r_{ijp} = 1$ if the route r and λ_i is the dual variable associated with constraint (6.2).

The dynamic programming algorithm 6.1 can be adapted to handle the multigrpah setting: a label at some node $i \in V^{MG}$ is extended along all arcs $(i,j)^p \in A^{MG}$ (instead of A in line 11 in Algorithm 6.1) and labels are only extended to customer and depot nodes (instructions in lines 21-25 in Algorithm 6.1 are not considered).

The used branching rule is similar to the standard one. If for any arc $(i,j)^p \in A^{MG}$ the flow is fractional, two branches are generated. In the first branch, the use of arc $(i,j)^p$ is

forbidden and in the second branch the arc $(i, j)^p$ must be used in the solution.

6.5 Computational experiments

In this section, we present the computational experiments carried out to evaluate the impact of the multigraph representation and the road network setting on the performance of the Branch-and-Price algorithm. We first present data tests used in the experiments in Section 6.5.1. In Section 6.5.2, we report the obtained results. A discussion on the obtained results is presented ins Section 6.5.3.

Branch-and-price algorithms are implemented in the C++ programming language. Tests are run on an Intel CORE i5 2.6 GHz computer with 4GB of memory. We use CPLEX 12.6 as the linear programming solver for restricted master problems.

6.5.1 Test data

In our experiments, we use three sets of instances: the first set of instances is provided by Letchford et al. [89], the second set of instances (*Letchford et al.-Like [89] instances*) are generated using the procedure proposed in [89] (described below) and the third set consists of instances derived using real data from the city of *Aix-in-Provence*, France.

6.5.1.1 Letchford et al. [89] and Letchford et al.-Like [89] (LL) instances

The first set of instances was generated by Letchford et al. [89] with the objective of simulating real-life road networks, using the following procedure:

1. Insert nodes at random positions in the Euclidean space;
2. Consider all possible arcs and insert new arcs sequentially (to represent road segments) on condition that the new inserted arc does not intersect with any other arc and has sufficiently large angles with other arcs at its endpoints;
3. Set the arc cost to the Euclidean distance between arc endpoints.

Using this procedure, Letchford et al. [89] generated different sparse graphs with different number of nodes n. In each graph, a node is randomly selected to be the depot location and other nodes are given a probability p to be customers. For each sparse graph, Letchford et al. [91] generated three different sets of travel times with different levels of correlation. These travel times are computed using $t_{ij} = \nu * c_{ij} + \mu * \gamma_{ij} * \bar{c}$ where γ_{ij} is a random number

in $[0, 1]$, $\bar{c} = \max_{(i,j)\in A} c_{ij}$, and parameters ν, $\mu \in [0, 1]$ are used to define the correlation degree between travel times and costs. Finally, they associate with each graph two m-TSPTW instances: a first instance with wide time windows (WTW) and a second instance with narrow time windows (NTW). Time windows are defined such that a set of routes, constructed in a greedy way, are feasible. An integer service time in $\{1, 2\}$ is defined for each customer node. We emphasize that only 6 instances with $n = 200$ and $p = 0.66$ were provided for us by Letchford et al. [89]: 2 instances with non-correlated travel times instances with weakly correlated travel times where and 2 instances with strongly correlated travel times. In order to be consistent with our problem definition, we define the vehicle capacity to 200 and we consider a fleet with a large number of vehicles. We assign a demand to each customer such that the routes defined by the time windows remain feasible.

Using the procedure described above, we generated the third set of instances with the same numbers of nodes and the same values of p as in [89]: $p = 0.66$ for $n \in \{50, 100, 150, 200, 250\}$ and $p = 0.33$ for $n \in \{100, 200, 300, 400, 500\}$. We also generated additional instances with different densities of customers for the road network with $n = 250$ and $n_c \in \{25, 50, 75, 100, 125\}$. Note that, in order to achieve a clear understanding, 5 different sparse graphs were generated for each configuration (n, n_c).

6.5.1.2 Real instances

The third set of instances is generated based on real data from the road network of the central urban area of the city of *Aix-en-Provence* (a city-commune in the region of Provence-Alpes-Cote d'Azur in the south of France, about 30 km north of Marseilles). Spacial data are extracted based on **OpenStreetMap©** [1] database. The considered road network is represented by a directed sparse graph with $n = 5437$ nodes and 10181 arcs where an arc represents a road segment and is defined by a length and a maximum allowed speed. Travel times are computed using road segments lengths and speeds. Costs are set as road segment lengths.

[1] OpenStreetMap is a collaborative project wich creates and distributes freely available geo-spatial data. www.openstreetmap.org/

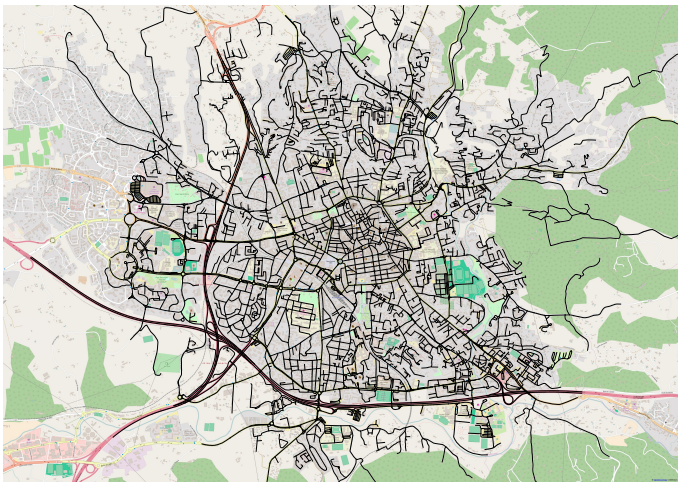


Figure 6.2: *Road Network of the central urban area of Aix-en-Provence*

Based on this road network, we generate 20 instances with $n_c \in \{5, 10, 25, 50\}$ (5 instances for each n_c). For each instance, depot and customer locations are randomly selected. Problem characteristics (time windows, customer demands, service times and vehicle capacity) are defined in the same way as for the first set of instances.

6.5.2 Results

In order to complete results presented by Letchford et al. [89] and to derive comprehensive conclusions, we propose the following experimental plan. In all cases, we compare the results obtained on the road network graph and on the multigraph:

1. We start by comparing results obtained by solving the LP relaxation using column generation where non-elementary routes are allowed for the m-TSPTW;
2. Then, we explore the case where only elementary routes are allowed. Note that in the m-TSPTW we do not consider customer demands and vehicle capacity constraints: the feasibility of a route is evaluated regarding only time window constraints;
3. Next, we investigate the impact of capacity constraints in the VRPTW on the performance of column generation algorithms;
4. Finally, we explore the complete branch-and-price algorithm.

In the first three steps of our experimental plan, we use the Letchford et al. [89] and LL instances. Real instances are used only for the complete branch-and-price algorithm. For all experiments, computing times are expressed in seconds and we limit the computing times to 7200 seconds.

Note that, we use the method described in [9] to generate multigraphs for all instances. We do not include multigraph construction time in the reported computing times.

6.5.2.1 Results for the multiple travelling Salesman Problem

Tables 6.1, 6.2 and 6.3 summarize results when solving the LP relaxation at the root node using column generation where non-elementary routes are allowed for the m-TSPTW. Tables 6.4, 6.5 and 6.6 summarize results where only elementary routes are allowed. In these tables, the first three columns indicate, respectively, the average number of nodes, the average number of arcs and the number of customers in road networks. Columns "corr" and "$|A^{MG}|$" indicate, respectively, the correlation level between travel times and costs, and the average number of arcs in the associated multigraph. For each type of instances (with narrow and wide time windows), we report the gap between the lower bound obtained with the multigraph representation (LB_{MG}) and the lower bound obtained on the road network (LB_{RN}) (column "Gap"), the average computing time on the multigraph (column "CPU_{MG}"), the average computing time on the road network (column "CPU_{RN}") and the ratio of the computing time on the multigraph to the computing time on the road network (column "$\frac{CPU_{MG}}{CPU_{RN}}$"). In this last column, we indicate the smallest ratio (column "Min"), the average ratio (column "Avg"), and the largest ratio (column "Max") over the 5 tested instances for each configuration (n, n_c and correlation level).

The gap between the lower bound obtained with the multigraph representation and the lower bound obtained on the road network is computed as follows:

$$Gap(\%) = \frac{LB_{RN} - LB_{MG}}{LB_{MG}} * 100 \tag{6.11}$$

Note that, this gap is due to the fact that, with the road network settings, a customer can be served multiple times consecutively in the same route, while in the multigraph, one needs to serve at least one intermediate customer. We do not report gaps for the case with only elementary routes as the same set of optimal routes are generated with both approaches, hence, the same value for the lower bound is obtained.

Table 6.1: *Results for column generation with non-elementary routes for the m-TSPTW on LL instances*

					Narrow Time Windows						*Wide Time Windows*					
					CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	Gap	*MG*	*RN*	Min	Avg	Max	Gap	*MG*	*RN*	Min	Avg	Max
50	135	33	NC	2362	-0.3%	0.3	0.4	0.56	0.83	1.22	-3.9 %	0.5	0.9	0.39	0.49	0.60
			WC	1864	-0.7%	0.4	0.5	0.43	0.67	0.96	-8.2 %	0.5	1.3	0.26	0.42	0.52
			SC	1266	-2.2%	0.3	0.4	0.45	0.65	0.91	-10.9%	0.5	1.3	0.33	0.41	0.48
100	278	66	NC	11795	-0.1%	1.4	1.4	0.72	1.18	2.08	-3.4 %	2.2	4.2	0.37	0.55	0.71
			WC	9581	0.0 %	1.6	1.9	0.63	0.88	1.16	-5.0 %	2.9	6.4	0.34	0.55	0.75
			SC	5265	-2.2%	1.8	2.3	0.51	0.79	1.03	-7.2 %	3.7	11.2	0.29	0.34	0.42
150	429	100	NC	32346	-0.1%	4.5	5.2	0.73	0.95	1.44	-3.7 %	9.3	19.7	0.38	0.55	0.80
			WC	25561	-0.1%	4.5	5.3	0.67	0.88	1.11	-5.2 %	10.4	19.5	0.41	0.55	0.72
			SC	13193	-2.6%	4.8	5.5	0.56	0.87	1.23	-6.2 %	11.3	27.8	0.35	0.41	0.44
200	574	133	NC	68979	-0.4%	10.9	11.7	0.81	0.95	1.12	-3.7 %	28.2	56.5	0.36	0.57	0.97
			WC	52742	-0.1%	12.1	13.5	0.72	0.94	1.32	-3.9 %	24.9	45.9	0.40	0.58	0.89
			SC	23798	-3.5%	12.0	12.2	0.51	1.05	1.79	-6.5 %	25.7	75.0	0.25	0.39	0.67
250	714	166	NC	116300	0.0 %	23.9	26.5	0.62	1.06	1.56	-2.4 %	44.1	62.4	0.55	0.79	1.02
			WC	92500	-0.1%	25.0	22.2	0.78	1.24	1.93	-3.9 %	44.4	71.8	0.48	0.65	0.85
			SC	37254	-2.1%	14.5	19.9	0.35	0.80	1.21	-5.4 %	50.1	135.3	0.32	0.39	0.55
100	278	33	NC	2994	-0.3%	0.2	0.5	0.29	0.41	0.50	-5.1 %	0.3	1.1	0.21	0.29	0.53
			WC	2400	-1.0%	0.2	0.5	0.36	0.44	0.56	-12.3%	0.3	1.7	0.14	0.22	0.35
			SC	1315	-3.8%	0.3	1.1	0.20	0.29	0.41	-19.2%	0.5	2.9	0.14	0.17	0.20
200	574	66	NC	16954	-0.1%	1.7	4.6	0.22	0.41	0.64	-6.3 %	2.2	12.5	0.16	0.18	0.24
			WC	13076	-0.1%	1.4	4.2	0.23	0.35	0.44	-11.2%	2.7	13.2	0.09	0.22	0.30
			SC	5806	-4.1%	1.2	5.3	0.19	0.23	0.29	-13.8%	2.2	21.3	0.07	0.11	0.15
300	869	100	NC	50418	-0.1%	6.8	11.9	0.43	0.57	0.71	-4.1 %	7.8	24.6	0.24	0.33	0.43
			WC	39084	-0.3%	4.5	9.9	0.35	0.47	0.59	-7.6 %	7.7	27.9	0.21	0.30	0.42
			SC	14545	-2.5%	3.7	12.8	0.22	0.31	0.45	-13.2%	7.5	68.9	0.09	0.12	0.18
400	1165	133	NC	103356	0.0 %	13.4	28.9	0.34	0.53	0.69	-7.3 %	25.1	80.0	0.25	0.35	0.52
			WC	77037	-0.1%	15.0	26.6	0.45	0.60	0.93	-9.8 %	23.7	100.4	0.16	0.27	0.34
			SC	26583	-4.2%	7.8	17.4	0.32	0.44	0.64	-12.1%	21.6	113.9	0.15	0.20	0.25
500	1458	166	NC	196267	-0.1%	38.8	56.7	0.61	0.73	1.06	-4.7 %	44.3	134.9	0.26	0.37	0.59
			WC	145782	-0.1%	30.1	75.1	0.31	0.55	0.97	-9.0 %	46.3	174.7	0.18	0.30	0.41
			SC	44751	-3.2%	20.5	62.0	0.19	0.40	0.58	-12.9%	32.8	271.4	0.09	0.13	0.17

Table 6.2: *Results for column generation with non-elementary routes for the m-TSPTW on LL instances with n = 250 nodes*

					Narrow Time Windows						*Wide Time Windows*					
						CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$				CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	Gap	*MG*	*RN*	Min	Avg	Max	Gap	*MG*	*RN*	Min	Avg	Max
250	714	25	NC	2553	0.0 %	0.1	1.1	0.09	0.14	0.19	-12.8 %	0.2	2.5	0.05	0.07	0.09
			WC	2123	0.0 %	0.1	1.0	0.11	0.15	0.19	-49.3 %	0.2	3.2	0.03	0.05	0.06
			SC	878	-17.7 %	0.1	1.4	0.08	0.09	0.10	-59.3 %	0.1	5.9	0.02	0.03	0.05
		50	NC	10704	-0.5 %	0.8	3.3	0.09	0.23	0.34	-8.0 %	1.0	7.4	0.11	0.14	0.19
			WC	8697	-0.1 %	0.7	3.2	0.11	0.25	0.37	-15.2 %	1.3	11.7	0.09	0.13	0.20
			SC	3478	-5.5 %	0.6	3.7	0.13	0.18	0.30	-21.5 %	1.3	16.0	0.05	0.08	0.12
		75	NC	24267	-0.2 %	2.4	5.9	0.22	0.41	0.64	-6.2 %	3.0	12.5	0.16	0.18	0.24
			WC	19333	0.0 %	1.8	5.5	0.23	0.35	0.44	-9.9 %	3.5	21.8	0.09	0.22	0.30
			SC	7815	-3.4 %	1.7	7.1	0.19	0.23	0.29	-17.5 %	3.9	41.8	0.07	0.11	0.15
		100	NC	42367	-0.3 %	5.8	9.3	0.34	0.43	0.56	-6.4 %	8.3	23.6	0.21	0.32	0.48
			WC	34163	0.0 %	4.9	9.5	0.23	0.34	0.53	-8.6 %	9.0	33.7	0.15	0.21	0.30
			SC	13716	-3.0 %	3.1	9.4	0.20	0.37	0.55	-10.9 %	9.9	63.3	0.15	0.18	0.22
		125	NC	66248	-0.2 %	10.5	13.3	0.47	0.62	0.84	-4.7 %	15.8	36.4	0.18	0.35	0.48
			WC	52534	-0.7 %	9.9	12.9	0.46	0.59	0.93	-6.0 %	18.8	45.0	0.20	0.31	0.46
			SC	21336	-1.5 %	7.4	12.9	0.39	0.63	0.89	-9.0 %	18.5	84.4	0.15	0.24	0.34
		166	NC	116300	0.0 %	23.9	26.5	0.81	0.95	1.12	-2.6 %	44.1	62.4	0.36	0.57	0.97
			WC	92500	-0.1 %	25.0	22.2	0.72	0.94	1.32	-4.2 %	44.4	71.8	0.40	0.58	0.89
			SC	37254	-2.2 %	14.5	19.9	0.51	1.05	1.79	-5.9 %	50.1	135.3	0.25	0.39	0.67

Table 6.3: *Results for column generation with non-elementary routes for the m-TSPTW on Letchford et al. [89] instances*

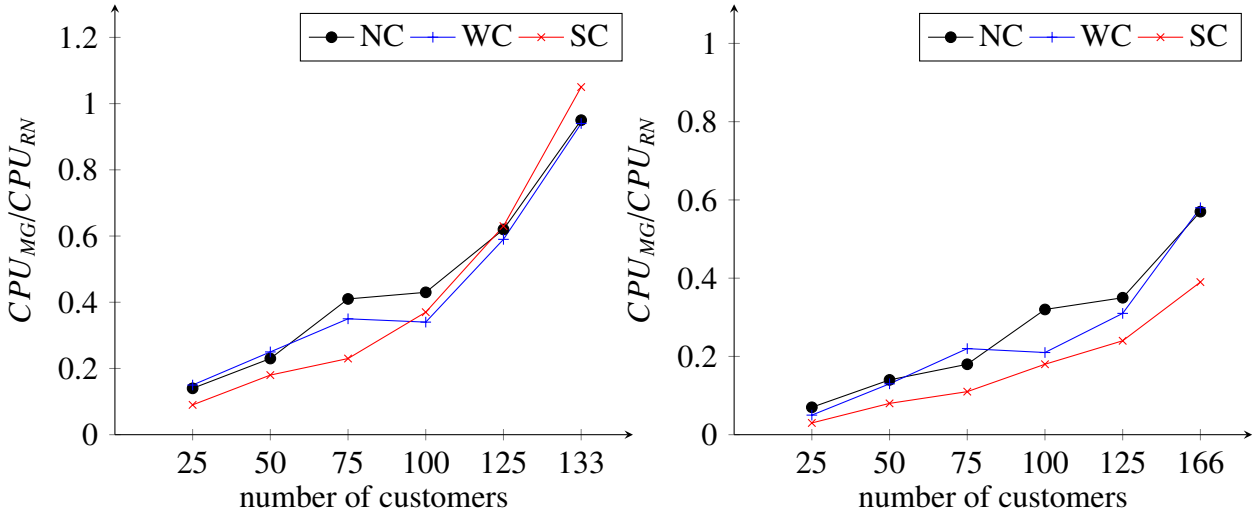
					Narrow Time Windows				*Wide Time Windows*			
						CPU (s)				CPU (s)		
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	Gap	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$	Gap	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$
200	133	582	NC	50368	-2.1 %	45.1	34.6	1.30	-4.3 %	54.3	43.7	1.24
			WC	38748	-1.3 %	16.2	17.9	0.91	-2.7 %	44.0	29.7	1.48
			SC	24758	-2.0 %	13.3	16.4	0.81	-2.3 %	31.2	27.8	1.12

Tables 6.1 and 6.2 show that a negative gap, which indicates that a lower value for LB_{RN} than for LB_{MG}, is obtained for almost all cases. This gap reaches -59.3% in average for strongly correlated instances with $n = 250$ nodes and $n_c = 25$ customers, and wide time windows. This negative gap is due to the fact that the pricing algorithm based on the road network permits a vehicle to serve a customer i, traverse some arcs (without servicing other customers) then return back to customer i and serve it again which is not allowed using the multigraph representation. From Table 6.1, we observe that for instances with the same n and n_c the difference between LB_{MG} and LB_{RN} becomes more important when the correlation between travel times and costs increases. We also observe that for the same correlation level this difference is more important when the time windows are wide. For LL instances with $n = 250$, Table 6.2, we see that the average gap between LB_{MG} and LB_{RN} decreases when the number of customers decreases, *e.g.*, for strongly correlated instances with wide time windows, it goes from -5.9% for $n_c = 166$ to -59.3% for $n_c = 25$.

From Tables 6.1 and 6.2, it comes out that the computing time increases when tackling the problem directly on the road network. The ratio of CPU_{MG} to CPU_{RN} is lower than 1 for the majority of instances. We mention that over the 450 tested instances, this ratio is greater than 1 for only 27 instances. The average ratio of CPU_{MG} to CPU_{RN} is higher for instances with narrow time windows and it increases when the correlation between travel times and costs decreases. For example, the average ratios for non correlated and strongly correlated instances with $n = 100$ and $n_c = 66$ are, respectively, 1.18 and 0.79 when time windows are narrow, and are 0.55 and 0.34 when time windows are wide. This behaviour can be explained by the fact that when time windows are wide, time constraints are less restrictive and thus, more label extensions are performed. However, this increase in the number of extensions is more important using the road network modelling since arcs represent short road segments with small travel times and many arcs need be traversed to reach a customer node. While an arc in the multigraph represents a path linking two customer, thus, it is easier to evaluate the feasibility of a route using the multigraph representation.

Another observation is that, for the same density of customers, the ratio of CPU_{MG} to CPU_{RN} increases when the number of nodes in the road network increases, *e.g.*, the ratio for NC instances with $n = 100$, $n_c = 33$ and narrow time windows is 0.29 in average and reaches 0.50, while it is 0.73 in average and reaches 1.06 for instances with $n = 500$ and $n_c = 166$. Figures 6.3 represents the evolution of the ratio of CPU_{MG} to CPU_{RN} with the number of customers for instances with 250 nodes for both type of time windows. We observe that by increasing the density of customers in the road network, the ratio of CPU_{MG} to CPU_{RN} increases. For example, for strongly correlated instances with narrow time windows, it ranges from 0.08 to 0.10 when $n_c = 25$, it ranges from 0.20 to 0.55 when $n_c = 100$ and it ranges from 0.51 to 1.79 when $n_c = 166$.

Table 6.3 reports the obtained results for Letchford et al. [89] instances. It comes that a similar behaviour is obtained as for LL instances. A negative gap between LB_{RN} and LB_{MG}



(a) *Instances with narrow time windows*

(b) *Instances with narrow time windows*

Figure 6.3: *Evolution of the ratio* $\frac{CPU_{MG}}{CPU_{RN}}$ *with the number of customers for instances with* $n = 250$

is obtained for the 6 tested instances. The ratio of CPU_{MG} to CPU_{RN} varies and it is more important when the correlation degree between travel times and costs is lower.

Table 6.4: *Results for column generation with only elementary routes for the m-TSPTW on LL instances*

					Narrow Time Windows					*Wide Time Windows*				
					CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	Min	Avg	Max	*MG*	*RN*	Min	Avg	Max
50	135	33	NC	2362	0.2	0.9	0.16	0.22	0.33	0.4	1.5	0.18	0.29	0.41
			WC	1864	0.2	1.7	0.09	0.12	0.21	0.6	2.1	0.21	0.28	0.47
			SC	1266	0.1	1.1	0.06	0.13	0.21	0.5	2.1	0.13	0.24	0.60
100	278	66	NC	11795	0.8	5.8	0.08	0.14	0.19	2.6	7.7	0.23	0.34	0.47
			WC	9581	0.9	6.5	0.10	0.14	0.21	5.6	10.9	0.28	0.52	0.76
			SC	5265	0.7	9.4	0.04	0.07	0.21	7.4	14.6	0.37	0.51	0.81
150	429	100	NC	32346	3.4	23.2	0.09	0.15	0.24	24.4	41.8	0.43	0.58	0.69
			WC	25561	3.5	29.1	0.10	0.12	0.14	26.8	42.4	0.42	0.63	0.83
			SC	13193	2.8	42.7	0.03	0.07	0.12	38.2	56.3	0.59	0.68	0.76
200	574	133	NC	68979	9.6	59.1	0.11	0.16	0.45	75.0	99.5	0.43	0.75	1.02
			WC	52742	7.5	74.2	0.08	0.10	0.22	85.5	96.0	0.40	0.89	1.60
			SC	23798	7.1	95.8	0.04	0.07	0.38	123.5	140.7	0.43	0.88	1.36
250	714	166	NC	116300	20.6	186.4	0.05	0.11	0.22	122.8	188.8	0.19	0.65	1.25
			WC	92500	20.2	138.5	0.13	0.15	0.19	165.5	211.8	0.24	0.78	1.24
			SC	37254	13.2	134.6	0.06	0.10	0.16	268.8	278.9	0.61	0.96	1.11
100	278	33	NC	2993.6	0.2	1.8	0.07	0.10	0.16	0.3	2.3	0.10	0.12	0.17
			WC	2399.6	0.2	1.9	0.07	0.10	0.19	0.3	3.1	0.09	0.10	0.11
			SC	1314.6	0.2	5.2	0.01	0.04	0.09	0.4	4.1	0.07	0.10	0.17
200	574	66	NC	16954	0.5	12.2	0.04	0.04	0.37	1.0	17.4	0.04	0.06	0.77
			WC	13076	0.3	15.9	0.01	0.02	0.51	1.3	18.4	0.03	0.07	0.69
			SC	5806	0.3	19.4	0.01	0.02	0.42	1.6	20.7	0.04	0.08	0.70
300	869	100	NC	50418	3.4	91.5	0.03	0.04	0.05	8.7	94.1	0.05	0.09	0.20
			WC	39084	2.8	67.7	0.02	0.04	0.06	11.7	100.6	0.05	0.12	0.24
			SC	14545	1.9	84.2	0.02	0.02	0.03	11.3	145.2	0.06	0.08	0.13
400	1165	133	NC	103356	11.1	169.7	0.05	0.07	0.08	38.6	239.7	0.11	0.16	0.29
			WC	77037	10.2	196.8	0.04	0.05	0.11	54.4	296.7	0.08	0.18	0.30
			SC	26583	5.9	144.4	0.03	0.04	0.05	58.0	267.6	0.12	0.22	0.34
500	1458	166	NC	196267	28.0	391.5	0.06	0.07	0.09	84.9	415.4	0.13	0.20	0.31
			WC	145782	25.1	417.5	0.06	0.06	0.07	112.9	656.0	0.12	0.17	0.20
			SC	44751	14.8	407.2	0.03	0.04	0.05	138.2	499.5	0.18	0.28	0.42

Table 6.5: *Results for column generation with only elementary routes for the m-TSPTW on LL instances with n = 250 nodes*

					Narrow Time Windows					*Wide Time Windows*				
					CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	Min	Avg	Max	*MG*	*RN*	Min	Avg	Max
250	714	25	NC	2553	0.1	6.9	0.01	0.02	0.02	0.1	8.3	0.01	0.02	0.02
			WC	2123	0.1	8.9	0.01	0.01	0.02	0.2	7.9	0.01	0.02	0.02
			SC	878	0.1	8.8	0.01	0.01	0.02	0.1	10.2	0.01	0.01	0.02
		50	NC	10704	0.4	17.6	0.02	0.02	0.03	1.1	23.5	0.02	0.05	0.09
			WC	8697	0.4	15.5	0.02	0.03	0.04	1.3	25.7	0.02	0.05	0.08
			SC	3478	0.3	20.6	0.01	0.02	0.03	1.2	33.4	0.03	0.04	0.06
		75	NC	24267	1.6	33.4	0.03	0.05	0.07	3.6	40.9	0.05	0.10	0.16
			WC	19333	1.3	33.6	0.04	0.04	0.04	4.1	44.0	0.07	0.09	0.11
			SC	7815	1.1	44.9	0.01	0.02	0.03	7.3	69.6	0.04	0.11	0.17
		100	NC	42367	3.5	63.9	0.03	0.06	0.08	10.7	79.7	0.06	0.14	0.23
			WC	34163	3.3	58.3	0.04	0.06	0.07	20.1	73.1	0.13	0.24	0.55
			SC	13716	2.5	60.8	0.03	0.04	0.07	25.9	97.3	0.20	0.26	0.34
		125	NC	66248	7.7	87.2	0.06	0.09	0.15	28.1	87.0	0.16	0.31	0.52
			WC	52534	7.3	101.3	0.05	0.08	0.10	52.2	127.2	0.24	0.41	0.53
			SC	21336	4.9	95.0	0.03	0.06	0.08	72.5	159.4	0.28	0.50	0.84
		166	NC	116300	20.6	186.4	0.05	0.14	0.22	122.8	188.8	0.19	0.67	1.25
			WC	92500	20.2	138.5	0.13	0.15	0.19	165.5	211.8	0.24	0.83	1.24
			SC	37254	13.2	134.6	0.06	0.11	0.16	268.8	278.9	0.61	0.94	1.11

Table 6.6: *Results for column generation with only elementary routes for m-TSPTW on Letchford et al. [89] instances*

					Narrow Time Windows			*Wide Time Windows*		
					CPU (s)			CPU (s)		
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$
200	133	582	NC	50368	75.8	175.0	0.43	456.5	265.1	1.72
			WC	38748	25.5	129.6	0.20	239.6	194.6	1.23
			SC	24758	19.7	101.6	0.19	199.2	132.7	1.50

From Tables 6.4-6.6, it comes out that for the multigraph based column generation where non-elementary routes are not allowed, the computing time increases for narrow time windows and decreases for wide time windows by decreasing the correlation degree. For the road network approach, increasing the correlation between the travel times and costs increases the average computing time for most cases with both types of time windows. We notice that the average ratio of CPU_{MG} to CPU_{RN} decreases when the correlation degree between travel times and costs increases, *e.g.*, for LL instances with $n = 200$ and $n_c = 133$ (Table 6.4), the ratio goes from 0.16 for NC instances to 0.07 for SC instances. Contrary to the case where non-elementary routes are allowed, the average ratio of CPU_{MG} to CPU_{RN} is more important when time windows are wide.

Another observation is that by allowing non-elementary routes the road network based column generation procedure is faster for almost all instances. While, using the multigraph representation, forbidding non-elementary routes decreases the computing time. For example, for NC instances with 500 nodes, 166 customers and narrow time windows, the average computing time on the multigraph goes from 28 seconds to 38.8 seconds by allowing non-elementary, while it decreases from 391.5 seconds to 56.7 seconds on the road network. Consequently, the ratio of CPU_{MG} to CPU_{RN} is reduced by restricting the column generation to the elementary routes. We mention that, the ratio of CPU_{MG} to CPU_{RN} is greater than 1 for only 12 out of 450 instances when only elementary routes are considered while it is greater than 1 for 47 instances when non-elementary routes are allowed.

Table 6.5 shows that increasing the number of customers for a fixed n increases the ratio of CPU_{MG} to CPU_{RN}. This ratio is 0.02 in average for NC instances with $n_c = 25$ and reaches 1.25 for NC instances with $n_c = 166$. We notice that, forbidding non-elementary routes reduces significantly the ratio of CPU_{MG} to CPU_{RN} when time windows are narrow. For example, for SC instances with $n_c = 125$ the ratio for the column generation procedure with non-elementary routes is 0.63 and reaches 0.89 while it is 0.06 in average and reaches 0.08 when these routes are forbidden.

Table 6.6 reports results Letchford et al. [89]. We notice that restricting the column generation procedure increases the computation time for both approaches, *e.g.*, for the WC instance with narrow time windows, CPU_{MG} and CPU_{RN} are respectively 16.2 and 34.6 when non-elementary routes are allows, and reach 25.5 and 129.6 seconds when these routes are not considered. We notice that by forbidding non-elementary routes and when time windows are narrow the increase in the CPU_{RN} is more important than the increase in the CPU_{MG}, consequently, the ratio $\frac{CPU_{MG}}{CPU_{RN}}$ is reduced. However, when time windows are wide, the increase in the CPU_{MG} is more important than the increase in the CPU_{RN}, thus, the ratio $\frac{CPU_{MG}}{CPU_{RN}}$ is higher when non-elementary routes are not allowed.

6.5.2.2 Results for the VRPTW

In this section, we present results obtained for the VRPTW. In this set of experiments, we only consider the pricing problem where non-elementary routes are not allowed. We first compare results obtained with the column generation at the root node of the search tree for the multigraph and the road network approaches (Tables 6.7, 6.8 and 6.9). Then, we compare results obtained for the complete branch-and-price schemes in Tables 6.10 - 6.13.

Table 6.7: *Results for column generation with only elementary routes for the VRPTW on LL instances*

					Narrow Time Windows					*Wide Time Windows*				
					CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	Min	Avg	Max	*MG*	*RN*	Min	Avg	Max
50	135	33	NC	2362	0.3	2.6	0.07	0.15	0.22	2.0	9.2	0.15	0.23	0.38
			WC	1864	0.4	5.4	0.06	0.08	0.10	3.9	14.3	0.13	0.21	0.38
			SC	1266	0.2	2.6	0.06	0.09	0.12	2.8	19.1	0.12	0.17	0.31
100	278	66	NC	11795	1.3	11.1	0.08	0.12	0.19	8.9	42.1	0.11	0.20	0.27
			WC	9581	1.7	20.5	0.06	0.09	0.10	16.5	82.1	0.16	0.19	0.22
			SC	5265	2.8	58.4	0.02	0.06	0.08	45.3	227.5	0.12	0.20	0.32
150	429	100	NC	32346	6.5	78.8	0.06	0.10	0.13	121.6	314.1	0.20	0.33	0.46
			WC	25561	6.0	90.0	0.04	0.07	0.09	116.5	318.9	0.28	0.34	0.50
			SC	13193	8.7	141.4	0.04	0.06	0.07	224.3	830.0	0.15	0.31	0.45
200	573.6	133	NC	68979	24.3	247.5	0.06	0.10	0.14	288.8	941.1	0.13	0.30	0.47
			WC	52742	22.5	292.4	0.05	0.08	0.12	367.7	1147.0	0.23	0.33	0.42
			SC	23798	28.4	412.0	0.04	0.08	0.11	963.4	2305.5	0.31	0.40	0.53
250	714	166	NC	116300	51.3	482.0	0.09	0.11	0.13	686.8	1371.7	0.23	0.40	0.68
			WC	92500	43.2	463.9	0.07	0.11	0.18	775.7	1428.0	0.30	0.50	0.70
			SC	37254	29.9	420.5	0.05	0.07	0.11	1218.7	3531.1	0.27	0.35	0.47
100	278	33	NC	2994	0.2	3.2	0.05	0.07	0.10	0.4	5.9	0.06	0.08	0.10
			WC	2400	0.2	3.9	0.05	0.06	0.10	0.5	10.4	0.03	0.07	0.14
			SC	1315	0.3	9.7	0.02	0.03	0.06	1.1	16.1	0.04	0.06	0.10
200	574	66	NC	16954	1.6	55.6	0.02	0.03	0.04	8.2	112.7	0.03	0.07	0.09
			WC	13076	1.5	45.9	0.03	0.03	0.04	11.8	156.0	0.05	0.08	0.12
			SC	5806	1.4	58.7	0.01	0.02	0.04	15.3	211.7	0.05	0.07	0.10
300	869	100	NC	50418	6.5	175.9	0.03	0.04	0.06	31.8	362.0	0.05	0.09	0.12
			WC	39084	4.8	166.1	0.02	0.03	0.04	28.9	466.1	0.03	0.09	0.14
			SC	14545	2.8	163.3	0.01	0.02	0.02	49.2	767.5	0.04	0.07	0.10
400	1165	133	NC	103356	19.5	458.4	0.03	0.05	0.06	179.2	1331.8	0.10	0.15	0.26
			WC	77037	22.3	540.2	0.03	0.05	0.06	208.2	1833.5	0.08	0.13	0.18
			SC	26583	10.7	315.1	0.03	0.04	0.04	215.7	1909.7	0.07	0.13	0.19
500	1458	166	NC	196267	49.9	1093.8	0.04	0.05	0.08	373.5	2801.9	0.12	0.14	0.17
			WC	145782	61.8	1697.3	0.03	0.04	0.04	821.8	6150.6	0.10	0.13	0.15
			SC	44751	22.9	1392.3	0.01	0.02	0.04	587.0	6190.8	0.10	0.11	0.14

Table 6.8: *Results for column generation with only elementary routes for the VRPTW on LL instances with n = 250*

					Narrow Time Windows					*Wide Time Windows*				
					CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	Min	Avg	Max	*MG*	*RN*	Min	Avg	Max
250	714	25	NC	2553	0.1	7.8	0.01	0.02	0.03	0.2	13.8	0.01	0.02	0.03
			WC	2123	0.2	12.1	0.01	0.02	0.02	0.3	15.1	0.02	0.02	0.03
			SC	878	0.1	9.3	0.01	0.01	0.02	0.2	18.9	0.01	0.01	0.03
		50	NC	10704	0.6	32.5	0.01	0.02	0.04	2.3	68.9	0.02	0.03	0.05
			WC	8697	0.7	31.8	0.02	0.02	0.03	3.5	129.5	0.02	0.03	0.04
			SC	3478	0.4	30.9	0.01	0.01	0.02	3.4	97.4	0.02	0.04	0.06
		75	NC	24267	2.4	78.2	0.02	0.03	0.04	8.8	164.7	0.04	0.06	0.09
			WC	19333	2.5	67.7	0.02	0.04	0.05	19.1	287.6	0.06	0.08	0.10
			SC	7815	1.6	76.2	0.01	0.02	0.03	22.3	320.4	0.04	0.07	0.10
		100	NC	42367	7.0	126.3	0.04	0.06	0.08	29.4	313.5	0.08	0.10	0.14
			WC	34163	6.4	133.1	0.03	0.05	0.06	86.2	588.6	0.11	0.14	0.16
			SC	13716	4.8	128.0	0.03	0.04	0.05	130.3	816.7	0.07	0.15	0.22
		125	NC	66248	19.8	245.0	0.05	0.08	0.10	92.8	538.4	0.12	0.17	0.23
			WC	52534	14.2	199.0	0.06	0.07	0.08	179.6	759.9	0.19	0.25	0.30
			SC	21336	9.3	180.4	0.04	0.05	0.06	255.4	1495.9	0.09	0.18	0.26
		166	NC	116300	51.3	482.0	0.09	0.11	0.13	686.8	1371.7	0.23	0.40	0.68
			WC	92500	43.2	463.9	0.07	0.11	0.18	775.7	1428.0	0.30	0.50	0.70
			SC	37254	29.9	420.5	0.05	0.07	0.11	1218.7	3531.1	0.27	0.35	0.47

Table 6.9: *Results for column generation with only elementary routes for VRPTW on Letchford et al. [89] instances*

					Narrow Time Windows			*Wide Time Windows*		
					CPU (s)			CPU (s)		
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$
200	133	582	NC	50368	327.4	1228.7	0.27	5071.6	3112.2	1.63
			WC	38748	76.5	787.8	0.10	975.6	1443.6	0.68
			SC	24758	37.2	469.3	0.08	720.4	1117.8	0.64

From tables 6.7 and 6.8 it comes out that, by considering customer demands and vehicle capacity constraints, the average computing time needed to column generation increases with both modelling approaches. This increase is more important on the road network than when using the multigraph representation, *e.g.*, for instances with narrow time windows, the CPU_{MG} is in average 3 times higher for the VRPTW than for the m-TSPTW and the CPU_{RN} is in average 6 times higher for the VRPTW on the road network. Consequently, the ratio $\frac{CPU_{MG}}{CPU_{RN}}$ is reduced by considering capacity constraints which indicates that the pricing algorithm for the VRPTW is much faster using the multigraph representation than on the road network.

We observe that for all cases, using the multigraph representation, the lower bound for branch-and-price scheme is computed in less computing time than using the road network approach. For all test-problems configurations, the ratio $\frac{CPU_{MG}}{CPU_{RN}}$ does not exceed 0.22 when time windows are narrow and 0.70 when time windows are wide. Table 6.8 shows that decreasing the density of customers in the road network decreases the difference between CPU_{MG} and CPU_{RN}, *e.g.*, for NC instances with narrow time windows, the multigraph based column generation procedure is 9.1 times faster than the column generation procedure on the road network when $n_c = 166$ and this acceleration factor reaches 50 when $n_c = 25$.

For Letchford et al. [89] instances, we see that the ratio of CPU_{MG} to CPU_{RN} is significantly reduced by considering capacity constraints. Except for NC instance with wide time windows, this ratio does not exceed 0.68 which means that the multigraph based column generation is at least 1.5 times faster than the column generation procedure that works directly on the road network.

Table 6.10: *Results for complete branch-and-price scheme for VRPTW on LL instances*

					Narrow Time Windows							*Wide Time Windows*						
					Solved		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		Solved		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	*MG*	*RN*	Min	Avg	Max	*MG*	*RN*	*MG*	*RN*	Min	Avg	Max
50	135	33	NC	2362	5	5	0.4	3.0	0.07	0.15	0.22	5	5	11.1	41.5	0.21	0.33	0.46
			WC	1864	5	5	0.8	12.3	0.05	0.08	0.13	5	4	132.0	386.0	0.03	0.25	0.49
			SC	1266	5	5	0.3	2.4	0.08	0.11	0.14	5	4	151.5	509.4	0.14	0.19	0.25
100	278	66	NC	11795	5	5	1.4	10.9	0.10	0.13	0.18	5	4	110.0	153.1	0.02	0.19	0.29
			WC	9581	5	5	3.3	141.2	0.01	0.10	0.16	4	3	216.9	529.6	0.14	0.16	0.20
			SC	5265	5	4	786.2	1311.2	0.00	0.04	0.07	0	0	7200.0	7200.0	-	-	-
150	429	100	NC	32346	5	5	7.3	77.4	0.06	0.11	0.15	1	1	314.0	2285.0	0.14	0.14	0.14
			WC	25561	5	5	12.1	112.9	0.04	0.09	0.15	1	0	4289.8	7200.0	-	-	-
			SC	13193	5	5	17.5	265.8	0.04	0.07	0.08	0	0	7200.0	7200.0	-	-	-
200	573.6	133	NC	68979	5	5	32.7	296.1	0.06	0.12	0.20	2	1	930.8	940.7	0.12	0.12	0.12
			WC	52742	5	4	123.0	1096.4	0.03	0.07	0.10	0	0	7200.0	7200.0	-	-	-
			SC	23798	5	4	208.9	754.1	0.06	0.10	0.15	0	0	7200.0	7200.0	-	-	-
250	714	166	NC	116300	5	3	183.1	1821.8	0.04	0.10	0.14	1	1	159.1	2258.3	0.07	0.07	0.07
			WC	92500	5	5	129.9	682.8	0.07	0.15	0.32	0	0	7200.0	7200.0	-	-	-
			SC	37254	5	3	1125.8	2270.5	0.06	0.08	0.09	0	0	7200.0	7200.0	-	-	-
100	278	33	NC	2994	5	5	0.2	2.5	0.08	0.10	0.12	5	5	1.4	162.3	0.00	0.08	0.12
			WC	2400	5	5	0.3	4.0	0.05	0.08	0.12	5	5	1.0	235.2	0.00	0.05	0.11
			SC	1315	5	5	0.3	7.9	0.03	0.05	0.08	5	5	2.7	142.8	0.02	0.06	0.10
200	574	66	NC	16954	5	5	2.2	58.2	0.02	0.03	0.05	5	4	40.9	352.4	0.02	0.07	0.11
			WC	13076	5	5	1.7	1106.5	0.00	0.03	0.04	3	3	53.8	1016.5	0.04	0.05	0.06
			SC	5806	5	5	2.1	63.6	0.01	0.03	0.07	4	2	149.4	6155.7	0.01	0.01	0.02
300	869	100	NC	50418	5	5	10.7	184.3	0.05	0.06	0.10	5	4	137.4	499.0	0.07	0.13	0.18
			WC	39084	5	4	8.7	213.0	0.04	0.05	0.05	3	1	256.8	1379.7	0.19	0.19	0.19
			SC	14545	5	5	5.9	405.5	0.01	0.02	0.03	3	1	1466.8	672.3	0.06	0.06	0.06
400	1165	133	NC	103356	5	4	66.4	1981.6	0.01	0.06	0.10	4	3	247.7	1499.4	0.11	0.15	0.20
			WC	77037	5	5	89.6	1338.5	0.04	0.07	0.12	3	0	4572.8	7200.0	-	-	-
			SC	26583	5	3	39.0	848.5	0.04	0.04	0.05	0	0	7200.0	7200.0	-	-	-
500	1458	166	NC	196267	5	3	102.8	1775.7	0.04	0.07	0.10	3	1	3005.8	2934.6	0.19	0.19	0.19
			WC	145782	5	4	241.4	1664.2	0.04	0.05	0.06	1	0	1643.0	7200.0	-	-	-
			SC	44751	5	3	56.5	4596.4	0.00	0.02	0.03	0	0	7200.0	7200.0	-	-	-

NOTE: *- indicates that no solution have been found within the time limit*

Table 6.11: *Results for complete branch-and-price scheme for VRPTW on LL instances with n = 250*

					Narrow Time Windows							*Wide Time Windows*						
					Solved		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		Solved		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	*MG*	*RN*	Min	Avg	Max	*MG*	*RN*	*MG*	*RN*	Min	Avg	Max
250	714	25	NC	2553	5	5	0.2	9.7	0.01	0.02	0.03	5	5	0.2	14.9	0.01	0.02	0.02
			WC	2123	5	5	0.2	12.4	0.01	0.02	0.02	5	4	0.4	20.1	0.01	0.02	0.03
			SC	878	5	5	0.1	11.7	0.01	0.01	0.02	5	5	0.3	21.9	0.01	0.02	0.04
		50	NC	10704	5	5	0.7	35.2	0.01	0.02	0.03	5	5	9.7	1167.8	0.01	0.03	0.06
			WC	8697	5	5	0.8	41.6	0.01	0.02	0.03	5	4	37.3	1219.3	0.02	0.03	0.03
			SC	3478	5	5	0.8	296.6	0.00	0.01	0.02	5	3	843.4	1080.7	0.02	0.04	0.04
		75	NC	24267	5	5	2.6	78.4	0.03	0.04	0.06	4	4	9.7	179.7	0.03	0.05	0.08
			WC	19333	5	5	2.4	71.9	0.03	0.04	0.04	3	3	13.0	151.3	0.08	0.08	0.10
			SC	7815	5	4	3.8	82.8	0.01	0.02	0.02	1	1	28.1	665.7	0.04	0.04	0.04
		100	NC	42367	5	5	12.5	312.4	0.03	0.06	0.11	5	4	106.8	562.8	0.06	0.16	0.25
			WC	34163	5	5	7.1	144.4	0.04	0.05	0.06	2	2	1505.9	2091.2	0.13	0.46	0.80
			SC	13716	5	4	274.7	197.9	0.02	0.85	3.33	0	0	7200.0	7200.0	-	-	-
		125	NC	66248	5	5	35.0	439.1	0.06	0.09	0.16	3	1	1285.8	241.1	0.14	0.14	0.14
			WC	52534	5	5	17.2	289.8	0.04	0.06	0.09	1	1	101.1	399.4	0.25	0.25	0.25
			SC	21336	5	4	10.8	217.3	0.03	0.04	0.05	0	0	7200.0	7200.0	-	-	-
		166	NC	116300	5	3	183.1	1821.8	0.04	0.10	0.14	1	1	159.1	2258.3	0.07	0.07	0.07
			WC	92500	5	5	129.9	682.8	0.07	0.15	0.32	0	0	7200.0	7200.0	-	-	-
			SC	37254	5	3	1125.8	2270.5	0.06	0.08	0.09	0	0	7200.0	7200.0	-	-	-

NOTE: *- indicates that no solution have been found within the time limit*

Table 6.12: *Results for complete branch-and-price scheme for VRPTW on Letchford et al. [89] instances*

					Narrow Time Windows			*Wide Time Windows*		
					CPU (s)			CPU (s)		
$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	$\|A^{MG}\|$	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$	*MG*	*RN*	$\frac{CPU_{MG}}{CPU_{RN}}$
200	133	582	NC	50368	767,5	4208,2	0,18	7200	7200	-
			WC	38748	143,4	7200,0	-	7200	7200	-
			SC	24758	7200,0	7200,0	-	7200	7200	-

NOTE: *- indicates that no solution have been found within the time limit*

Tables 6.10-6.12 report results for the complete branch-and-price scheme for LL and Letchford et al. [89] instances. Columns "Solved" indicate the number of instances solved within the time limit (7200 seconds) using the multigraph representation and on the road network.

From Tables 6.10-6.11, we observe that, when time windows are narrow, all instances are solved using the multigraph representation and 134 out of 150 instances are solved directly on the road network. For instances with wide time windows, 78 and 57 instances are solved respectively on the multigraph and on the road network. We also observe that, the average computation times for both approaches increase when extending customers time windows. For example, for NC instances with $n = 100$, $n_c = 150$, and with narrow time windows, the average computation times are 32.7 and 296.1 seconds respectively for the multigraph representation and the road network approach and reach 930.8 and 940.7 seconds when time windows are wide. We notice that by increasing the density of customers in the road network, the number of solved instances decreases and the average computation times increase for both multigraph and road network approaches. The increase in computation time is more important when tackling the problem directly on the road network. Thus, the ratio of CPU_{MG} to CPU_{RN} decreases when the density of customers in the road network increases.

From Table 6.12 it comes out that using the multigraph representation, two instances are solved within the time limit, while an optimal solution has been found for only one instance when tackling the problem directly on the road network. The ratio $\frac{CPU_{MG}}{CPU_{RN}}$ for the instance solved using both approaches is equal to 0.18 which indicates that the multigraph-based branch-and-price algorithm is more than 5 times faster than the branch-and-price algorithm working directly on the road network.

Table 6.13 reports results for real instances. Columns "Sav_{MC}" and "Sav_{MT}" indicates the savings on solution cost by using the multigraph representation or the road network compared to respectively the cheapest-path-based graph and the fastest-path-based graph. We introduce these columns to illustrate the impact of considering alternative paths (through the multigraph representation or by solving the problem directly on the original road network) on the solution quality. We see that all real instances are solved within 18 seconds using the multigraph representation while no optimal solution has been found within the time limit for 2 instances when tackling the problem directly on the road network. The ratio of CPU_{MG} to CPU_{RN} does not exceed 0.016 which means that the multigraph based branch-and-price algorithm is at least 62.5 times faster than the algorithm that works directly on the road network.

Table 6.13 shows that the solution quality is significantly improvement by considering all alternative paths. The obtained saving reaches 17.6% compared to solution on cheapest-path-based graph and reaches 17.4% compared to solution on fastest-path-based graph.

Table 6.13: *Results for the complete Branch-and-price algorithm with instances on the road network of Aix*

$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$		CPU_{MG}	CPU_{RN}	$\frac{CPU_{MG}}{CPU_{RN}}$	Sav_{MC}	Sav_{MT}
5437	10181	5	1	0.05	5.48	0.009	17.6 %	7.3 %
			2	0.07	4.16	0.016	12.0 %	4.7 %
			3	0.05	4.28	0.011	0.0 %	17.4 %
			4	0.06	8.77	0.007	0.0 %	11.8 %
			5	0.05	14.84	0.003	0.0 %	9.4 %
		10	1	0.09	11.87	0.008	8.7 %	7.0 %
			2	0.08	7.08	0.011	6.2 %	5.8 %
			3	0.07	6.03	0.011	0.0 %	11.9 %
			4	0.09	11.57	0.007	0.5 %	6.8 %
			5	0.08	25.65	0.003	0.0 %	11.5 %
		25	1	0.20	56.85	0.004	3.8 %	7.3 %
			2	0.20	51.35	0.004	1.9 %	6.0 %
			3	0.17	35.10	0.005	6.5 %	7.9 %
			4	0.39	111.84	0.004	5.1 %	13.0 %
			5	0.18	80.95	0.002	11.2 %	9.5 %
		50	1	0.99	113.46	0.009	5.6 %	16.2 %
			2	3.33	7200.00	-	1.1 %	7.4 %
			3	2.11	147.63	0.014	1.1 %	6.3 %
			4	1.03	252.03	0.004	8.5 %	10.8 %
			5	17.36	7200.00	-	5.5 %	5.5 %

6.5.3 Discussion

The computational study presented in this paper shows that it is difficult to confirm that a road network based branch-and-price scheme is more efficient than using the a multigraph representation (as stated by Letchford et al. [89]). Numerical results show that the relative efficiency of the two approaches depends on several factors. Allowing non-elementary routes in the pricing problem could be advantageous for the road network approach in term of computing time, however, this could have a negative impact on the quality of the lower bound and thereafter could lead to a longest branching time. Increasing the density of customers in the road network could be also advantageous for the road network approach. Another factor is the wideness of customer time windows. We observe that when time windows are less restrictive, the growth in the number of feasible label extensions on the road network is more important than when using the multigraph representation. Thus, the acceleration factor of the multigraph-based column generation compared to the road network based column generation is larger when time windows are wider. We show also that considering customer demands and vehicle capacity constraints has an effect on the relative efficiency of both approaches. Experiments on the m-TSPTW and on the VRPTW show that for the same instances, considering capacity constraints increases the computing times, especially on the road network and therefore decreases the ratio of the computing time with the multigraph representation to the computing time on the road network.

The obtained results show that for most part of test problems, it is more interesting to tackle the problem using the multigraph representation which is in contrast with the conclusions drawn by Letchford et al. [89]. This might be due to many reasons. First, Letchford et al. [89] generate only one route with a negative reduced cost at each iteration of the column generation, while in our experiments all non-dominated routes with negative reduced cost are generated at each iteration. Note that in a branch-and-price scheme, it is more efficient to generate all columns with negative routes at each iteration of the column generation. Second, in order to achieve a clear understanding and to avoid side effects, we propose to base our experiments on a large set of instances with 5 instances for each configuration (number of nodes, number of customers, correlation level and time windows width). Then, we report results with average values. While, computational experiments in [89] are based on only one instance for each configuration which may lead to limited conclusions. In addition, Letchford et al. [89] consider instances with relatively high densities which is not the case for real life applications. Results obtained on instances with several densities and real instances contradicts Letchford et al. [89] conclusions and show that it is more interesting to tackle the problem using the multigraph representation. Finally, by comparing our results to those reported in [89], we observe that computing times for the multigraph approach obtained by Letchford et al. [89] are extremely high which might be due to the nature of data structures that are not very adapted to the multigraph setting.

6.6 Conclusion

Vehicle routing problems have been intensively studied in the operational research literature. Most of proposed approaches are based on a key assumption that the best path between a pair of customer nodes can be easily defined. Thus, the problem can be tackled by representing the road network with a customer-based graph of the road network. In many real life applications, several attributes have to be defined on road segments. In this case, several alternative paths with different compromises could exist between each pair of nodes. Not considering these alternative paths could have a negative impact on the solution quality. An increasing number of papers in the literature investigate this issue and propose two alternative approaches. In some of them, a multigraph representation is used to handle alternative paths. Others propose to solve the problem directly on the original road network. A key paper in this regard is presented by Letchford et al. [89]. Authors present a numerical comparison between these two approaches and show that a column generation procedure that works directly on the road network is more efficient.

In this paper, we investigate more in depth the road network based approach and propose to complete results presented in [89]. We propose a complete branch and price that can handle the road network setting. We conduct a computational experiments based on instances from the literature and instances derived from real road network data. We analyse the impact of different problem characteristics, such as customer density in the road network, time and capacity constraints on the efficiency of the branch and price algorithm with both the multigraph representation and the road network modelling. Obtained results show that in most cases it is more interesting to tackle the problem using the multigraph representation.

Future works would investigate vehicle routing problems with several attributes on road segments and where travel times are time dependent. In this case, computing and handling the multigraph representation may be intractable since the set of non-dominated paths between each pair of nodes depend on the departure time and may vary over the time. Consequently, enumerating all these paths for every possible departure time could be very difficult and representing the road network with a multigraph representation could be unmanageable. For these reasons, we think that it is more suitable to solve time dependent vehicle routing problems directly on the road network.

Chapter 7

The Time-Dependent Vehicle Routing Problem with Time Windows and road network information

This chapter is a working paper.

Abstract

In the literature, most approaches proposed to solve time-dependent vehicle routing problems assume that for each pair of interest points (e.g., depot, customers...), a travel-time function is known. Almost no paper investigates how these functions can be computed from travel times in the underlying road network. Furthermore, most of them neglect the possibility that different paths could be selected in the road network depending on the compromises they offer between cost (distance) and travel-time. In this paper we propose the first exact solution approach for these problems that starts from travel-time function expressed at the level of the road network. Computational study carried out on realistic instances and on instances derived from a real road-network illustrate the important impact of the proposed modeling on solution values.

Keywords: *Time Dependent Vehicle Routing Problem, Branch-and-Price, Road Network.*

7.1 Introduction

Vehicle routing problems define a class of combinatorial optimization problems that aim at computing minimum-cost routes for a fleet of vehicles in order to serve a set of customer locations in a road network. Each customer is served exactly once and each route starts and ends at a central depot. The total demand delivered along a route should not exceed the vehicle capacity. Due to their numerous application, vehicle routing problems have been widely studied in the operations research literature (see *e.g.*, [87], [65], [128]). Most of proposed

approaches assume that the travel time between two customer locations is invariant over the time. In real life applications, this assumption does not hold. Travel times and speeds are subject to significant variations over the day. These variations may be due to predictable events such as traffic congestion and unpredictable events such as accidents, weather conditions and vehicle breakdowns. In the literature, these variations are addressed using time-dependent variants of vehicle routing problems (see *e.g*, [57], [84], [43], [54]). In such variants, with each road segment is associated a time function that indicates the travel time for every departure date.

Typically, vehicle routing problems, including those introducing time-dependent travel times, are tackled using a so-called customer-based graph, where a node is introduced for every point of interest (depot and customer locations) and arcs represent the best paths linking these points. This approach relies on the assumption that the best path between two points in the original road network can be easily defined (at each possible starting time for time-dependent problems). In many situations, this assumption is not valid [10]. One of these situations is when several attributes are defined on road segments. In this case, alternative paths with different compromises could exist between two points in the road network. Discarding these alternatives when tackling the vehicle routing problem can have a negative impact on the solution quality [61, 8].

In the literature, several papers have investigated the negative impact of the customer-based graph on solution quality for non-time-dependent vehicle routing problems (VRPs). It has been shown that transforming the original road network into a customer-based graph could result in important losses of optimality [8]. Two approaches have been proposed to handle this issue. The first approach consists in representing the road network with a multigraph. In this representation, all efficient paths between two points of interest are considered and maintained when solving the problem [61, 8, 7, 85]. The second approach consists in tackling the problem directly on a graph that mimics the original road network, called the road-network graph [89].

In this paper, we propose to address this issue for time-dependent VRPs. Intuitively, it is all the more essential to avoid customer-based graphs in this case because time-dependent VRPs better capture congestion effects and should show more contrasted alternative paths. We select the Time-Dependent Vehicle Routing Problem with Time Windows (TDVRPTW) as test-bed problem.

Similarly to non-time-dependent VRPs, two modeling approaches are *a priori* available. However, due the time-dependency, it is difficult or even intractable to represent the road network with a multigraph. Indeed, one would have to compute the set of efficient paths for each pair of points of interest at each possible departure time, which induces the solution of many NP-hard problems (see [121], [67]) and the use of complex data structures. For this reason, we propose to address the problem on a road-network graph. We coin the problem as

TDVRPTW$_{RN}$. In this problem, a road-network graph is given, with travel-speed functions assigned to every arc. We develop a branch-and-price algorithm able to solve exactly the TDVRPTW$_{RN}$. We compare our solutions with those found on two customer-based graphs obtained from the road-network graph: a min-cost graph where paths are selected according to their travel distance, a min-time graph where paths are selected according to their travel time. We base our experiments on two type of instances: (1) benchmark instances simulating small real-life road networks; (2) instances derived from a large real-world road network.

It is important to highlight than details on how travel-time information can be obtained for customer-based graphs are missing in most papers on time-dependent VRPs. Conventionally, in these papers, the customer-based graph is introduced first. Then, time-varying speed profiles are introduced on every arc (of the customer-based graph). These speed profiles are finally converted into travel times (see *e.g.*, [73], [76], [27]). With this approach, speeds or travel times are not explicitly defined on road network arcs. Furthermore, it implicitly assumes that the speed is constant on all the road-network arcs that compose an arc in the customer-based graph.

Only a few papers are interested in how travel speed and time functions can be obtained from road-network information. Several of these studies are not concerned with vehicle routing. They aim at computing point-to-point fastest paths for a given starting time (see *e.g.*, [35], [102], [80]). A stream of papers examines the issue of constructing a database for travel times on time-dependent road networks. Eglese et al. [47] propose a model that constructs a Road Timetable based on historical data and provides the shortest travel time for different departure times. They illustrate the benefits of considering the Road Timetable information when tackling a vehicle routing problem. The main drawback of their approach is that fastest-paths are computed only for a set of specified departure times. Then, the travel time for any departure time is approximated using these computed values. This procedure may thus provide a weak estimation of travel times, with consequences on solution quality or feasibility.

In order to correctly evaluate travel-time functions for our two customer-based graphs, we develop two specific algorithms. The two algorithms start with the same information as that of the VRPTW$_{RN}$, *i.e.*, travel-speed functions associated with arcs of the road-network graph.

In the rest of this paper, we first review the relevant literature in Section 7.2. In Section 7.3, we formally describe the TDVRPTW$_{RN}$, introduce some mathematical background (definitions and notation) and detail the two algorithms developed to construct the two customer-based graphs. In Section 7.4, we present the branch-and-price algorithm developed for the solution of the TDVRPTW$_{RN}$. Finally, we present computational experiments and comment results in Section 7.5.

7.2 Literature review

Despite the huge number of papers dealing with VRPs, the number of papers addressing time-dependent problems is relatively limited. As far as we know, the first study dealing with a vehicle routing problem where travel times vary over time is by Beasley [6]. Interested readers can refer to [62] for a recent survey and for a discussion on the different models and problems that arise from time-dependent data.

Time-dependent VRPs are most of the times tackled on customer-based graphs. In the literature, an increasing number of papers investigate the negative impact that this graph can have on solution quality for non-time-dependent vehicle routing problems. Garaix et al. [61] were the first to point out that transforming road-network information into a customer-based graph can result in losing solution optimality. They propose to replace the customer-based graph with a multigraph in which all efficient paths are maintained. This modeling approach is investigated more in depth by Ben Ticha et al. [8]. They propose a branch-and-price algorithm and present an extensive computational study based on instances from the literature and instances representing real road networks. Reported results confirm the negative impact of the customer-based graph. Letchford et al. [89] suggest that applying the branch-and-price framework to a road-network graph would be more efficient. However, they limit their investigations to the pricing problem. More recently, Ben Ticha et al. [11] revisited the results presented by Letchford et al. [89]. They propose a complete branch-and-price scheme and conduct extensive comparisons between the multigraph and the road-network graph approaches. Their results contradict those presented in [89] and conclude that in most cases the multigraph-based branch-and-price algorithm is more efficient.

The aforementioned papers confirm that when several attributes are defined on road segments, the traditional customer-graph can have negative effects on solution quality for VRPs. In the time-dependent VRPs literature, little attention has been paid to this issue. The most relevant study is proposed by Huang et al. [71]. They introduce the so-called Time Dependent Vehicle Routing Problem with Path Flexibility (TDVRP-FP). The Path Flexibility means that when solving the problem a set of alternative paths between each pair of nodes are maintained in a multigraph-like structure. These alternative paths present different compromises in terms of travel time and distance and are computed as follows: a modified Dijkstra's algorithm is first applied to compute the fastest paths for a set of discretized departure time then the shortest path is included into the set of considered paths. Huang et al. [71] propose a mathematical formulation for the TDVPR-FP where the selection of a path to be used to link every two customer nodes is embedded as a decision variable. They present a computational study based on instances derived from the road network of Beijing, China. Numerical results show that the path flexibility improves significantly the solution quality in terms of cost (up to 5%) and fuel consumption (up to 7%). The author observe that these improvements are more significant than those obtained using flexible departure times at the depot and customer nodes. They also investigate the TDVRP-FP under stochastic traffic conditions and notice

that the path flexibility provides a natural recourse in this case.

Setak et al. [122] also introduce a multigraph to tackle a problem that they call Time Dependent Pollution Routing Problem. In their model, parallel arcs present different compromises on travel times, energy consumption and tolls cost. But, they do not specify how these arcs are computed. To solve the problem, they propose a tabu search heuristic. Their computational experiments show that, using the multigraph, an average gain of 1.1% is achieved on solution costs compared to costs obtained with a customer-based graph.

Besides the papers cited above, similar modelings have been used but with relatively different goals. Setak et al. [123] investigate a time-dependent vehicle routing problem defined on a multigraph where parallel arcs represent different travel speed distributions and satisfy the FIFO property. Wang and Lee [133] introduce the so-called Time-Dependent Alternative Vehicle Routing Problem, where a time window is associated with each node and every pair of nodes is linked with two arcs. The first arc is defined with a time-dependent travel-speed distribution and is used when traffic is low. The second arc is defined with a constant travel time and could be used as an alternative during peak hours. In both studies, the aim of considering parallel arcs between two nodes is not to provide different compromises but to provide different alternatives to be used depending on the departure time: at any point in time, one arc dominates the others.

We notice that there is only a few studies that investigate time-dependent VRPs with travel-time information defined at the level of the road network. Among these studies, even less investigate the impact of transforming the road network into a customer-based graph. Furthermore, they all lose some information in the process: they all introduce what can be called heuristic multigraphs, that is, multigraphs where not all efficient paths are present. This limit is consistent with what was claimed in Section A.1, *i.e.*, that representing the road network with a multigraph is hardly tractable when travel-times are time-dependent.

It is astonishing that no paper propose an approach that can ensure the optimality of solutions. This an important contribution of this paper, thus allowing, as a primary objective, to evaluate the optimality gaps due to the use of customer-based graphs.

7.3 Preliminaries

We first describe, in Subsection 7.3.1, the TDVRPTW$_{RN}$ and the travel-time functions used in this paper. In Subsection 7.3.2, we present basic concepts and operators that are mainly used for the construction of the customer-based graphs, as described in Subsections 7.3.3 and 7.3.4.

7.3.1 Problem description

The Time-Dependent Vehicle Routing Problem With Time-Windows and Road-Network Information (TDVRPTW$_{RN}$) is defined on a directed road-network graph $G^{RN} = (V^{RN}, A^{RN})$. V^{RN} contains the depot node 0 and nodes that represent road junctions. Among these nodes, a subset C represents customers. An arc $(i, j) \in A^{RN}$ models a road segment and is defined with a travel-distance (road-segment length) d_{ij} and a travel-time function $\tau_{ij}(t)$ which depicts the time needed to go through arc (i, j) for a given departure time at node i. Every customer $i \in C$ is assigned a time-window $[e_i, l_i]$, a service time st_i and a demand q_i. The time horizon $[0, H]$ is given by the depot time-window $[e_0, l_0]$. A fleet $\mathcal{K}$ of homogeneous vehicles with a finite capacity Q is available to perform the deliveries.

The TDVRPTW$_{RN}$ consists in designing a set of optimal routes for the fleet of vehicles. The total load delivered along a route cannot exceed the vehicle capacity. Each customer has to be served exactly once and within its time window. Customer nodes can however be traversed several times, as any other road junctions, without service.

Travel-time functions τ_{ij} are obtained from speed profiles. Speed profiles are defined by step-wise functions that divide the planning horizon $[0, H]$ into p_{ij} time zones. For each time zone, the travel speed is constant on arc (i, j). We denote by $T_{ij}^0, T_{ij}^1, ..., T_{ij}^{p_{ij}}$ the time zone boundaries with $T_{ij}^0 = 0$ and $T_{ij}^{p_{ij}} = H$, and by s_{ij}^k the travel speed on time zone $[T_{ij}^k, T_{ij}^{k+1}[$. T_{ij}^k are also called *speed breakpoints* because speed changes occur at these points.

Based on its speed profile, the travel-time function $\tau_{ij}(t)$ on an arc (i, j) can be calculated using the procedure described in [73]. The main idea in this procedure is that the travel-speed is not necessarily constant over the entire length of the road segment, as it changes when the boundary between two consecutive time periods is crossed. The resulting travel-time function is a continuous piecewise linear function and can be represented by the coordinates of its *travel-time breakpoints*. As showed in [73], such travel-time function satisfies the FIFO property, *i.e.*, a later departure at node i results in a later arrival at node j.

The quantity $a_{ij}(t) = t + \tau_{ij}(t)$ gives the arrival time at node j given a departure time t at node i. $a_{ij}(t)$ is a piecewise linear non-decreasing function and can be described using coordinates at *arrival time breakpoints* which are directly deduced from *travel time breakpoints*. In this paper, we use arrival time functions as this leads to simpler formulations.

7.3.2 Backgrounds and basic operations

In this section, we present basic notation and operators used mainly for the computation of customer-based graphs (presented in Sections 7.3.3 and 7.3.4).

Let f and g be two arrival time functions. f and g are piecewise linear and non-decreasing functions defined on the time horizon $[0, H]$. f can be described using the following informations:

- $N(f)$: number of linear pieces into which f is divided.
- t_f^k: right boundary of the time interval on which is defined the k^{th} piece of f (with also $t_f^0 = 0$).
- $\alpha(f, k)$: linear coefficient of the k^{th} piece of f.
- $\beta(f, k)$: constant term in the k^{th} piece of f.

Using these notations, f can be defined as follows;

$$f(t) = \begin{cases} f^{(1)}(t) & if\ t_f^0 \leq t \leq t_f^1 \\ f^{(2)}(t) & if\ t_f^1 \leq t \leq t_f^2 \\ \dots & \\ f^{(N(f))}(t) & if\ t_f^{N(f)-1} \leq t \leq t_f^{N(f)} \end{cases}$$

with

$$f^{(k)}(t) = \alpha(f, k)t + \beta(f, k)$$

Alternatively, f can be represented using the set of interpolation points

$$I(f) = \{(t_f^0, w_f^0), (t_f^1, w_f^1), \dots, (t_f^{N(f)}, w_f^{N(f)})\}$$

where $w_f^k = f(t_f^k)$.

Assuming that f and g are associated with two successive arcs (in this order), the arrival time at the end of the second arc in function of the starting time at the beginning of the first arc is given by the composition of f and g, denoted $g \circ f$. It should be clear that $g \circ f$ is a piecewise linear function as it involves two piecewise linear functions. To to define this function we have to compute its set of interpolation points $I(g \circ f)$. This is can be done as follows

- With each interpolation point (t_f^k, w_f^k) of f is associated an interpolation point $(t_f^k, g(w_f^k))$ for $g \circ f$.
- With each interpolation point (t_g^l, w_g^l) of g is associated an interpolation point $(t_{f^{-1}}^l, w_g^l)$ for $g \circ f$ where $t_{f^{-1}}^l$ is such that $f(t_{f^{-1}}^l) = t_g^l$. Note that $t_{f^{-1}}^l$ is easily computed once the index k such that $w_f^k \leq t_g^l < w_f^{k+1}$ is found: $t_{f^{-1}}^1 = \frac{t_g^l - \beta(f,k)}{\alpha(f,k)}$.

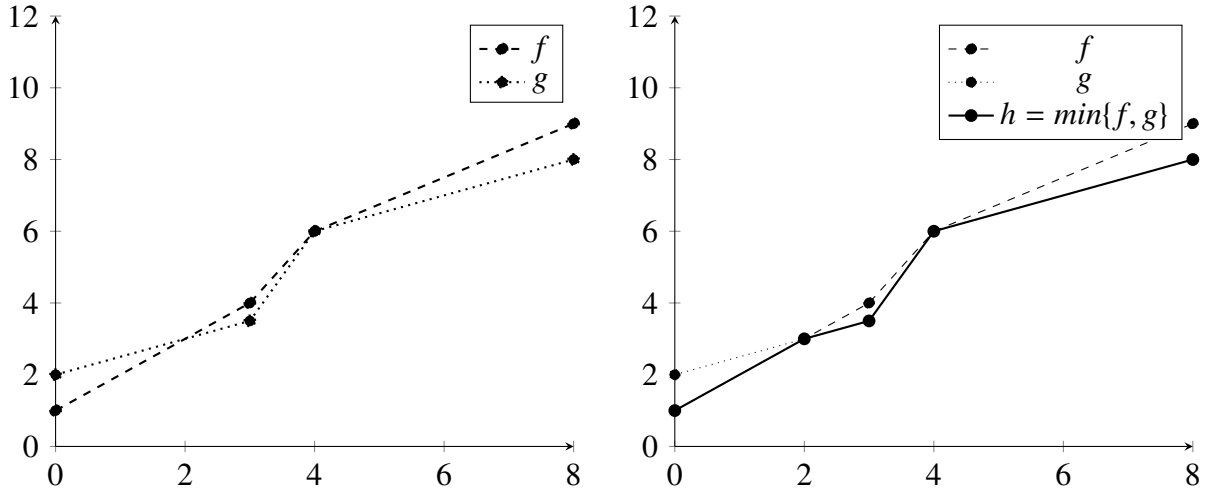


Figure 7.1: *Illustration of* $min\{f, g\}$

$I(g \circ f)$ is the union of these two set of points.

Let us now consider two arrival time functions f_1 and f_2. We say $f_1 < f_2$ if $f_1(t) < f_2(t)$ for all $t \in [0, T]$. Similarly, we say $f_1 \leq f_2$ if $f_1(t) \leq g_1(t)$ for all $t \in [0, T]$.

Another important operator that we need to define is $min\{f_1, f_2\}$ which returns the function $t \mapsto min\{f_1(t), f_2(t)\}$ for all $t \in [0, H]$. Figure 7.1 gives an illustration of the operator $min\{f_1, f_2\}$.

Let (f_1, g_1) and (f_2, g_2) be two couples of arrival time functions. We say that we *merge* g_1 and g_2 in g according to $min\{f_1, f_2\}$ and we note $g = \underset{min\{f_1,f_2\}}{merge}\{g_1, g_2\}$ if $g(t) = g_1(t)$ for each t such that $min\{f_1, f_2\}(t) = f_1(t)$ and $g(t) = g_2(t)$ for each t such that $min\{f_1, f_2\}(t) = f_2(t)$. An illustration of the operator $\underset{min\{f_1,f_2\}}{merge}\{g_1, g_2\}$ is presented in Figure 7.2. As will be seen, this operator is used to compute the distance associated with a travel-time function. If f_1 and f_2 represent the travel time functions for two different paths linking the same nodes i and j, and g_1 and g_2 represent the length of these paths, $min\{f_1, f_2\}$ indicates the best travel time that can be obtained from these two paths and $\underset{min\{f_1,f_2\}}{merge}\{g_1, g_2\}$ the associated travel distance.

In the next two sections, we present the two algorithms used to compute the min-cost (customer-based) graph and the min-time (customer-based) graph.

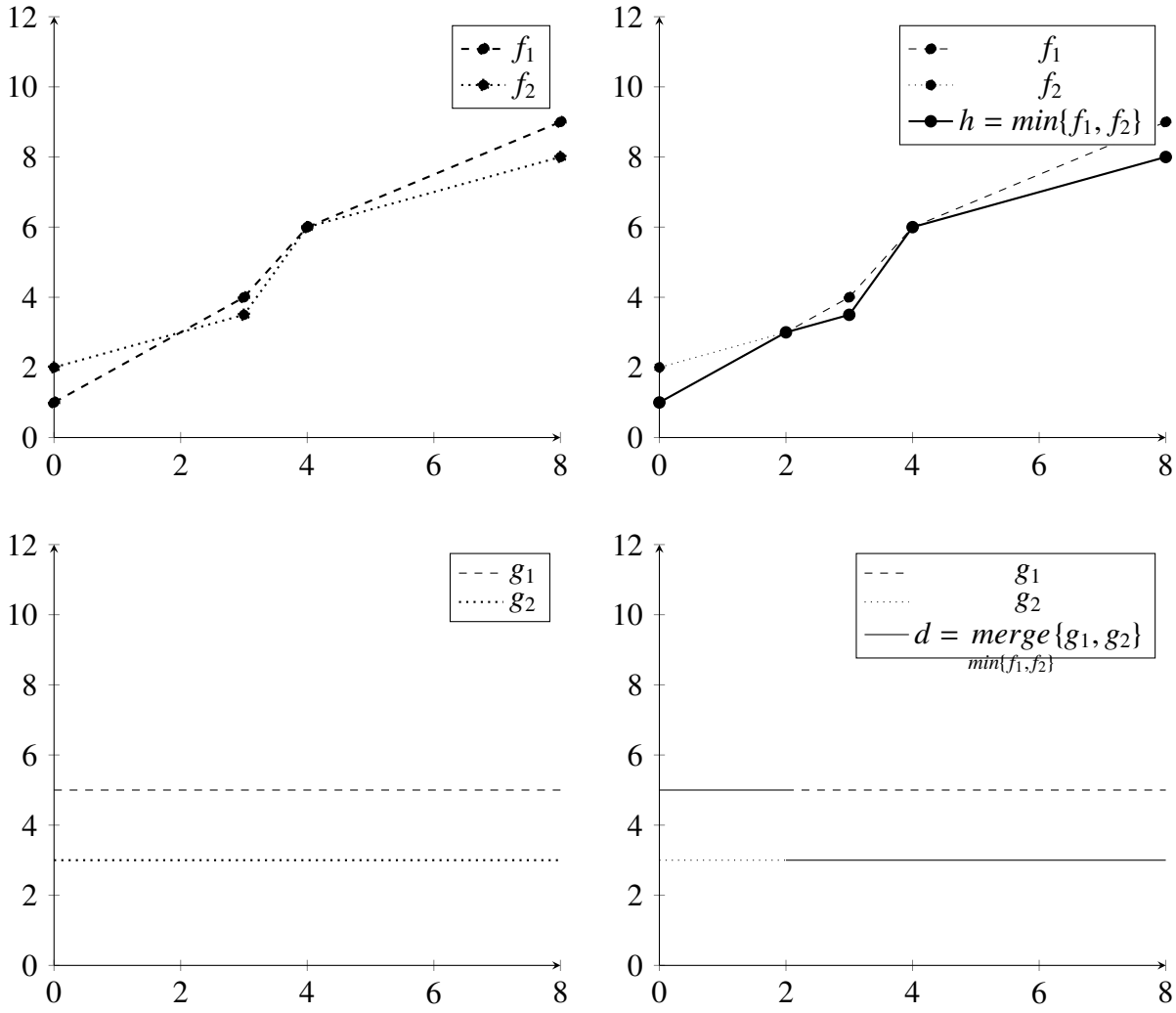


Figure 7.2: *Illustration of* $merge_{min\{f_1,f_2\}}\{g_1, g_2\}$

7.3.3 Time-dependend shortest path algorithm

In the min-cost graph, arcs represent the shortest paths (in distance) linking points of interest (depot and customer locations). All these paths can be computed easily by applying $|C| + 1$ times Dijkstra's algorithm in G^{RN}, successively starting from the depot and the customer nodes. However, for each path, we also need to compute the associated arrival time function. This can be introduced in Dijkstra's algorithm as shown by Algorithm 7.1.

Algorithm 7.1 Algorithm for time-dependent shortest path

1: $d_{ss}^{SP} \leftarrow 0$
2: $a_{ss}^{SP} \leftarrow \hat{0}$
3: **for all** $i \in V^{RN} \setminus \{s\}$ **do**
4: $\quad d_{si}^{SP} \leftarrow \infty$ and $a_{si}^{SP} \leftarrow \widehat{\infty}$
5: **end for**
6: $Q \leftarrow V^{RN}$ //priority queue of nodes sorted according to d_{si}^{SP}
7: **while** Q is not empty **do**
8: $\quad i = Q.extractMin()$
9: $\quad$ **for all** $j \in V^{RN}$ such that $(i, j) \in A^{RN}$ **do**
10: $\quad\quad d = d_{si}^{SP} + d_{ij}$
11: $\quad\quad$ **if** $d < d_{sj}^{SP}$ **then**
12: $\quad\quad\quad d_{sj}^{SP} \leftarrow d$
13: $\quad\quad\quad a_{sj}^{SP} \leftarrow a_{ij} \circ a_{si}^{SP}$
14: $\quad\quad\quad Q.updatePriority(j, d_{sj}^{SP})$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end while**
18: **return** d_{si}^{SP} and a_{si}^{SP} for all $i \in C \cup \{0\}$

The best subpath from the source node $s \in \{0\} \cup C$ to a node $i \in V^{RN}$ is labeled with two values d_{si}^{SP} and a_{si}^{SP} where :

- d_{si}^{SP} represents the length of the subpath;
- a_{si}^{SP} represents the arrival time function at node i, starting at s and following this subpath.

Algorithm 7.1 is identical to Dijkstra's algorithm except for three lines: 2, 4 and 13.

Line 2 initializes to $\hat{0}$ the arrival time function at the source node, where $\hat{0}(t) = 0$, $\forall t \in [0, H]$. Line 4 initializes to $\hat{\infty}$ the arrival time function for all other nodes, where $\hat{\infty}(t) = +\infty$, $\forall t \in [0, H]$. Line 13 updates the arrival time function when a label is modified, *i.e.*, a subpath has been improved. As explained in Section 7.3.2, the new function is computed by the composition of the function associated with the preceding node and the function

associated with the last arc in the subpath.

Algorithm 7.1 has a pseudo-polynomial complexity $O(nN^*)$ where $n = |C| + 1$ and $N^* = \sum_{(i,j)\in A^{RN}} N(a_{ij})$.

Proof. Operator $f \circ g$ requires $O(N(f)+N(g))$ operations. The selection of a node (Line 8) and the update of the priority of a node (Line 14) require $O(log(n))$ operations each. Thus, an iteration of the while loop (Line 7-17) is performed in $O(log(n)+\sum_{j/(i,j)\in A^{RN}} (N(a_{si}^{SP}) + N(a_{ij}) + log(n)))$, with i the selected node. Since every node in V^{RN} is selected exactly once, the while loop is performed in total in

$$O\left(\sum_{i\in V^{RN}}\left(log(n) + \sum_{j/(i,j)\in A^{RN}} (N(a_{si}^{SP}) + N(a_{ij}) + log(n))\right)\right)$$

$$= O\left(nlog(n) + \sum_{(i,j)\in A^{RN}} (N(a_{si}^{SP}) + N(a_{ij}) + log(n))\right)$$

Because the shortest path between s and any node i is the composition of arrival time functions for a given subset of arcs in A^{RN}, we have $N(a_{si}^{SP}) \leq N^*$ and the above complexity can be changed to

$$O\left(nlog(n) + |A^{RN}|N^* + N^* + |A^{RN}|log(n)\right)$$

Finally, $n \leq |A^{RN}|$ and $|A^{RN}| \leq N^*$ and, because G^{RN} is a sparse graph, $|A^{RN}| = O(n)$. Therefore, Algorithm 7.1 runs in $O(nN^*)$ operations in the worst case. □

7.3.4 Time-dependend fastest path algorithm

In the min-time graph, arcs represent the fastest paths linking points of interest (depot and customer locations). They are described by two functions: an arrival time function and a distance function. Note that a distance function is needed because the fastest path may change depending on the departure time.

The paths can be computed by applying $|C| + 1$ times a labeling algorithm in G^{RN}, providing for a given source node in $s \in C \cup \{0\}$ the fastest paths to all destination nodes in G^{RN} for all possible departure times. This algorithm is depicted in Algorithm 7.2. It is based on a label correcting procedure where node labels are defined by two functions: the arrival time function a_{si}^{FP} and the distance functions d_{si}^{FP}.

These functions are computed using the operators presented in Section 7.3.2. The main idea of the algorithm is that if the arrival time function f obtained by extending the subpath at a node i along an arc (i, j) improves the arrival time function at node j for at least one

$t \in [0, H]$, *i.e*, there exists $t \in [0, H]$ such that $f(t) < a_{sj}^{FP}(t)$, then the algorithm updates a_{sj}^{FP} to $min\{f, a_{sj}^{FP}\}$. Furthermore, as it means that at that time t a different path is preferred, d_{sj}^{FP} is modified to $\underset{min\{f, a_{sj}^{FP}\}}{merge}\{d_{si}^{FP} + d_{ij}, d_{sj}^{FP}\}$.

Algorithm 7.2 Algorithm for time-dependent fastest path

1: $d_{ss}^{FP} \leftarrow \hat{0}$
2: $a_{ss}^{FP} \leftarrow \hat{0}$
3: **for all** $i \in V^{RN} \setminus \{s\}$ **do**
4: $\quad d_{si}^{FP} \leftarrow \widehat{\infty}$ and $a_{si}^{FP} \leftarrow \widehat{\infty}$
5: **end for**
6: $Q \leftarrow V^{RN}$ // priority queue of nodes sorted according to $a_{si}^{FP}(0)$
7: **while** Q is not empty **do**
8: $\quad i \leftarrow Q.extractMin()$
9: $\quad$ **for all** $j \in V^{RN}$ such that $(i, j) \in A^{RN}$ **do**
10: $\quad\quad f \leftarrow a_{ij} \circ a_{si}^{FP}$
11: $\quad\quad$ **if not** $a_{sj}^{FP} \leq f$ **then**
12: $\quad\quad\quad a_{sj}^{FP} \leftarrow min\{f, a_{sj}^{FP}\}$
13: $\quad\quad\quad d_{sj}^{FP} \leftarrow \underset{min\{f, a_{sj}^{FP}\}}{merge}\{d_{si}^{FP} + d_{ij}, d_{sj}^{FP}\}$
14: $\quad\quad\quad$ **if** $j \in Q$ **then**
15: $\quad\quad\quad\quad Q.updatePriority(j, a_{sj}^{FP}(0))$
16: $\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad Q.insert(j, a_{sj}^{FP}(0))$
18: $\quad\quad\quad$ **end if**
19: $\quad\quad$ **end if**
20: $\quad$ **end for**
21: **end while**
22: **return** d_{si}^{FP} and a_{si}^{FP} for all $i \in C \cup \{0\}$

Contrary to Algorithm 7.1, the maximal number of iterations of the while loop (Lines 7-21) cannot easily be determined. It might actually be possible that, given two nodes s and i, an exponential number of paths linking these two nodes are, at some instant t, the faster. In this case, arrival time function a_{si}^{FP} (and the associated function d_{si}^{FP}) would be updated an exponential number of times. Hence, Algorithm 7.2 would require an exponential number of operations.

7.4 Branch-and-price algorithm for the TDVRPTW$_{RN}$

In this section, we present the branch-and-price scheme used to solve the TDVRPTW$_{RN}$. The motivation for using such a solution approach is that it has performed very well for many transportation problems and has become one of the most efficient exact solution methods to solve vehicle routing problems.

The algorithm proposed in this paper is based on the branch-and-price scheme presented in [11] for the same problem with constant travel times. For this reason, we briefly describe the main components of the algorithm. Compared to [11], two main modifications have been carried out. First, the composition operator described above is used in the pricing problem when extending labels. Second, bidirectional dynamic programming has been implemented. It is worth mentioning that our objective in this paper is not to achieve the best possible implementation of the branch-and-price algorithm for the TDVRPTW$_{RN}$, but it is to develop a solution method that allows deriving comprehensive conclusions.

The branch-and-price algorithm is based on the Dantzig-Wolf decomposition [32]. This decomposition consists in a specific problem reformulation that gives rise to an integer master problem with a tighter linear relaxation than the compact formulation of the problem. However, this master problem involves a large number of variables, thus, it cannot be solved using a standard branch-and-bound procedure. To handle this issue, a column generation procedure is used, resulting in a branch-and-price algorithm.

7.4.1 Master Problem

The master problem for the TDVRPTW$_{RN}$ consists in set covering formulation (7.1)-(7.4):

$$Min \sum_{r\in\Omega} c_r x_r \tag{7.1}$$

$$s.t \quad \sum_{r\in\Omega} a_{i,r} x_r \geq 1 \quad \forall i \in C \tag{7.2}$$

$$\sum_{r\in\Omega} x_r \leq K \tag{7.3}$$

$$x_r \in \{0,1\} \quad \forall r \in \Omega \tag{7.4}$$

where Ω denotes the set of feasible routes, c_r is the cost of route $r \in \Omega$ and $a_{ir} = 1$ if customer $i \in C$ is served in the route r, 0 otherwise. Binary variable x_r takes value 1 if the route r is selected in the optimal solution, 0 otherwise.

In the branch-and-price scheme, a restriction of the LP relaxation of (7.1)-(7.4) to a subset of routes $\Omega_1 \subset \Omega$, the so-called restricted Master Problem ($MP(\Omega_1)$), is solved at each itera-

tion. Ω_1 is composed of all routes generated at previous iterations and is iteratively enlarged by solving the pricing problem. For a detailed tutorial on the branch-and-price scheme, we refer the reader to [52].

7.4.2 Pricing problem

The pricing problem aims at finding new routes offering better ways to serve customers, *i.e.*, with negative reduced costs. The pricing problem can be reduced to a Time Dependent Shortest Path Problem with Resource Constraints (TDSPPRC) in G^{RN}. Ben Ticha et al. [11] notice that, in the road-network graph, the elementary path condition has to be modified. Indeed, in an optimal solution an arc of the road network can be traversed several times and a customer node can be visited several times. In what follows, we say that a route or a path is elementary if customer nodes in the route are served at most once, even if they are traversed several times.

To solve the TDSPPRC, we develop a time-dependent labeling algorithm which is a modification of the labeling algorithm proposed in [11]. Basically, this algorithm performs as follows. A subpath from the depot to a node $i \in V^{RN}$ is represented using a label defined by $L = (i, t, c, q, S)$ where t is the arrival time at node i, c is the reduced cost associated with the subpath, q is the total demand of customers served along the subpath and S represents the set of customers that are not reachable anymore, either because they have already been served or because they cannot be reached due to resource constraints. At each iteration, all labels at a certain node $i \in V^{RN}$ are extended along arcs $(i, j) \in A^{RN}$. When the destination node j is a customer node, two extensions are processed. In the first extension, the service at the customer j is performed, if it is feasible. In the second extension, the node is only visited without servicing the associated customer. To handle the exponentially growing number of generated labels, we use the dominance rule proposed by Feillet et al. [53] and defined as follows:

Definition 7.1. *A label $L_1 = (v, t_1, c_1, q_1, S_1)$ dominates a label $L_2 = (v, t_2, c_2, q_2, S_2)$ if:*

1. $t_1 \leq t_2$
2. $c_1 \leq c_2$
3. $q_1 \leq q_2$
4. $S_1 \subseteq S_2$

In order to improve the efficiency of the pricing algorithm, we implement a bi-directional search strategy. To do this, we associate two labels with each node i: a forward label defined as described above and, a backward label representing the subpath from the node i to the depot and defined by $L_B = (i, t, c, q, S)$ where t is the latest starting time at node i, c is the reduced cost associated with the subpath, q is the total demand of customers served along

the subpath and S represents the set of unreachable customers. Forward labels are extended to the middle of the planning horizon and not further, while backward labels are extended and are allowed to cross the middle of the planning horizon. At the end, forward labels and backward labels arriving at the same node are merged to generate a complete route.

Note that the efficiency of the labeling algorithm depends on the length of the subpaths defined by the labels. The bi-directional search strategy aims at limiting the length of generated labels and thus, improves oftenly the computing time. More details on the bi-directional search can be found in [114] and [30].

7.4.3 Branching scheme

The branching rule aims at eliminating the current fractional solution by adding constraints and partitioning the solution space. As shown in [11], the standard branching rule used for vehicle routing problems is not suitable in a road-network graph. The reason is that in customer-based graph every arc is traversed at most once in a feasible solution. This property does not hold anymore in a road-network graph.

Ben Ticha et al. [11] propose the following branching scheme:

- Select an arc $(i, j) \in V^{RN}$ with a fractional flow ϕ_{ij} given by $\phi_{ij} = \sum_{r\in\Omega_1} b_{ijr}x_r$ where b_{ijr} represents the number of times arc (i, j) is traversed along route r;
- Derive two branches:
  - In the first branch, set the flow upper limit on arc (i, j) to $\lfloor\phi_{ij}\rfloor$
  - In the second branch, set the flow lower limit on arc (i, j) to $\lfloor\phi_{ij}\rfloor + 1$

To address this branching rule in the column generation procedure, constraints (7.5) or (7.6) are added to the restricted master problems:

$$\sum_{r\in\Omega_1} b_{ijr}x_r \leq \lfloor\phi_{ij}\rfloor \tag{7.5}$$

$$\sum_{r\in\Omega_1} b_{ijr}x_r \geq \lfloor\phi_{ij}\rfloor + 1 \tag{7.6}$$

At the pricing problem level, the cost associated with arc (i, j) is set to $c_{ij} = c_{ij} - \lambda_{ij}^{low}$ in the first branch and to $c_{ij} = c_{ij} - \lambda_{ij}^{up}$ in the second branch where $\lambda_{ij}^{low} \leq 0$ and $\lambda_{ij}^{up} \geq 0$ are the dual variables associated with constraints (7.5) and (7.6), respectively.

As underlined by Ben Ticha et al. [11], this branching rule does not guarantee the feasibility of the solution. Indeed, an integer arc flow may correspond to a fractional routing solution. To handle this issue, a specific procedure is proposed. Again, we derive two branches:

- In the first branch, we check if graph $\tilde{G} = (V^{RN}, \tilde{A})$, where $\tilde{A} = \{(i, j) \in A : \sum_{r \in \Omega_1} b_{ijr} \tilde{x}_r > 0\}$, contains a feasible solution. This is done by first enumerating all the feasible routes in $\tilde{G}$, then solving the resulting set covering formulation with an integer programming solver.
- In the second branch, we enforce the use of at least one arc that is not traversed by the fractional solution $\tilde{x}$. For this aim, we add constraint $\sum_{(i,j) \in A \setminus \tilde{A}} \sum_{r \in \Omega_1} b_{ijr} x_r \geq 1$ to the restricted master problem and we subtract the associated dual variable $\tilde{\lambda}$ from the cost of every arc $(i, j) \in A \setminus \tilde{A}$ when solving the pricing problem.

7.5 Computational experiments

In this section, we present the experimental study carried out. We first present the benchmark problems used. Then, we summarize the obtained results and we evaluate the impact of tackling the problem directly on the road network instead of using a customer-based graph.

We implement the branch-and-price algorithm in the C++ programming language and we use CPLEX 12.6 as the linear programming solver for restricted master problems. Tests are run on an Intel CORE i5 2.6 GHz computer with 8GB of memory. We limit computing times for branch-and-price algorithms with different modelings to 7200 seconds.

7.5.1 Test data

In our experiments, we use two sets of instances: the first set of instances (*Letchford et al. [89]-like* instances) are generated with the objective to simulate real road networks and the second set of instances are derived from a real road network.

7.5.1.1 Letchford et al.[89]-like instances

The Letchford et al.[89]-like (LL) instances are generated using the procedure proposed in [89] and that performs as follows. First, it randomly inserts nodes in the Euclidean space, then it considers all possible arcs and insert a new arc in A^{RN} when it does not provoke any intersection with other arcs and angles with other arcs at its endpoints are large enough. Arc costs are set according to Euclidean distances.

Table 7.1: *Speed factor profiles*

	z_1	z_2	z_3	z_4	z_5
Congestion-free road segments	1.5	1	1.67	1.17	1.33
Normal road segments	1.17	0.67	1.33	0.83	1
Congestion-bound road segments	1	0.33	0.67	0.5	0.83

With this procedure, we generate three sparse graphs with $n = |V^{RN}| \in \{50, 100, 200\}$ nodes. A node is randomly selected to represent the depot location and remaining nodes are given a probability p to be customers. For the sparse graph with $n = 50$, we generate instances with $n_c = 16$ and $n_c = 33$ customers. For the sparse graph with $n = 100$, we generate instances with $n_c = 25$, $n_c = 33$ and, $n_c = 50$ customers. For the sparse graph with $n = 200$, we generate instances with $n_c = 25$ and $n_c = 50$ customers.

For each sparse graph, three different sets of static travel times with different levels of correlation are computed using formula $t_{ij} = \nu * c_{ij} + \mu * \gamma_{ij} * \bar{c}$ where $\bar{c} = max\{c_{ij} : (i, j) \in A^{RN}\}$, γ_{ij} is a random number in $[0, 1]$ and, ν and μ are correlation parameters defined in $[0, 1]$. The first set of static travel times has non-correlated travel times (NC), the second set has weakly correlated travel times (WC), the third set has strongly correlated travel times (SC). These travel times represent "nominal" travel times in the road network during non-congested periods and are used to compute "nominal" travel speeds.

To take into account traffic congestion, we associate with each road segment different road profiles. We divide the planning horizon $[0, H]$ into 5 periods: $z_1 = [0, 0.2H[$, $z_2 = [0.2H, 0.3H[$, $z_3 = [0.3H, 0.7H[$, $z_4 = [0.7H, 0.8H[$ and, $z_5 = [0.8H, H[$. The second and the fourth periods are congestion periods. Speeds in the remaining periods are relatively high. Implicitly, with this decomposition, we assume that for all road segments, speed breakpoints are the same. It makes sense in practice since congestion tends to happen around the same time periods (peak hours) in all the road network. We assume that there are three types of road segments: congestion-bound, normal and congestion-free. The type of the road segment is randomly defined and remains the same for the different correlation levels on each road network. Speed on a road segment is then obtained by multiplying its nominal speed by a factor depending on both the segment type and the period, and reported in Table 7.1.

Finally, we associate with every road network two TDVRPTW$_{RN}$ instances: the first instance with narrow time windows (NTW) and the second instance with wide time windows (WTW). Time windows are defined such that a set of routes, constructed in a greedy way without considering congestion, are feasible. An integer service time in $\{1, 2\}$ is defined for each customer node. We set the vehicle capacity to 200 and we consider a fleet with a large number of vehicles. We assign a demand to each customer such that the routes defined by the time windows are feasible.

7.5.1.2 Real instances

The second set of instances are generated based on real data from the road network of the central urban area of the city of *Aix-en-Provence* (a city-commune in the region of Provence-Alpes-Cote d'Azur in the south of France, about 30 km north of Marseilles). Spacial data are extracted based on **OpenStreetMap©** [1] database. The considered road network is represented by a directed sparse graph with $n = 5437$ nodes and 10181 arcs where an arc represents a road segment and is defined by a length and a maximum allowed speed. Costs are set as road segment lengths.

Time dependency parameters (time periods, speed profiles, road segments types) are defined as described in 7.5.1.1. However, the type of the road is defined based on the maximum allowed speed given in data. For highways, motorways and arterial roads (characterized with high maximum allowed speeds), the road segment type is set to "normal". For streets, boulevards and roads in the center of the city, the road segment type is set to "congestion-bound". For small roads and living streets (characterized with low maximum allowed speeds), the road segment type is set to "congestion-free".

Based on this road network, we generate instances with $n_c \in \{5, 10, 25\}$, with three instances for each value of n_c. For each instance, depot and customers locations are randomly selected. Time windows, customer demands, service times and vehicle capacity are defined in the same way as for the first set of instances.

7.5.2 Results

In order to investigate the impact of solving the TDVRPTW using the complete road-network information, we compare the obtained solutions to the solutions computed when tackling the problem with the min-cost graph and the min-time graph. Note that for real instances we only compare the solutions of the TDVRPTW$_{RN}$ to the solutions obtained on the min-cost graph. The reason is that, due to the complexity of the proposed algorithm (see Section 7.3.4), it is highly time consuming to generate an exact min-time graph in a large road network.

Table 7.2 presents the computing times required by the construction of the min-cost and min-time graphs for LL instances. Tables 7.3 and 7.4 compare the results of the branch-and-price algorithms for LL instances. Similar information for real instances is presented in Table 7.5. Columns "$|V^{RN}|$", "$|A^{RN}|$" indicate the number of nodes and the number of arcs in the road-network graph. Column "$|C|$" indicates the number of customers. Column

[1] OpenStreetMap is a collaborative project wich creates and distributes freely available geo-spatial data. www.openstreetmap.org/

Table 7.2: *Results for the construction of simple graph representations for Letchford et al. [89]-Like instances*

$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	construction min-cost(s)	construction min-time(s)
50	134	16	NC	0.2	2.0
			WC	0.2	1.5
			SC	0.2	2.1
		33	NC	0.4	4.5
			WC	0.4	4.0
			SC	0.4	4.9
100	286	25	NC	0.5	10.8
			WC	0.5	10.1
			SC	0.5	14.7
		33	NC	0.8	17.3
			WC	0.8	16.1
			SC	0.8	22.5
		50	NC	1.4	29.1
			WC	1.5	26.8
			SC	1.4	34.9
200	580	25	NC	1.3	46.7
			WC	1.3	46.9
			SC	1.3	80.4
		50	NC	3.4	138.7
			WC	3.4	113.0
			SC	3.6	152.1

"*Corr*" gives the correlation degree between travel times and costs. Columns "*construction min-cost graph(s)*" and "*construction min-time graph(s)*" report the computing times (in seconds) for the construction of the customer-based graphs. Column "Ins" indicates the instance index for real instances. Columns "*CPU min-cost(s)*", "*CPU min-time(s)*" and "*CPU Road Network(s)*" report the computing times of the branch-and-price algorithms, expressed in seconds. The "gap" columns report the gap between solution values of the TDVRPTW$_{RN}$ and solution values in the associated customer-based graph. This gap is computed as follows:

$$gap(\%) = \frac{cost\ in\ road\ network\ graph - cost\ in\ customer\ based\ graph}{cost\ in\ customer\ based\ graph} * 100 \qquad (7.7)$$

As expected, we see from Table 7.2 that the computing time needed for the construction of the min-cost graph is more important than the computing time of the min-cost graph, and that it increases significantly with the number of customers.

From Tables 7.3 and 7.4, it comes out that 38 out of 42 TDVRPTW$_{RN}$ instances are solved, while 32 and 37 optimal solutions only are obtained when using the min-cost graph and the min-time graph, respectively. The branch-and-price algorithm for the TDVRPTW$_{RN}$ is faster for 14 out of 31 solved instances compared to the min-cost graph and is faster for 14

Table 7.3: *Results for Letchford et al. [89]-Like instances with narrow time windows*

$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	CPU min-cost(s)	CPU min-time(s)	CPU Road Network(s)	gap min-cost(%)	gap min-time(%)
50	134	16	NC	**Infeasible**	3.8	95.6	–	**-19.8**
			WC	10.9	4.6	1.7	0.0	**-9.6**
			SC	1.2	0.8	1.0	0.0	**-1.3**
		33	NC	1.2	31.9	3.5	–	**-14.2**
			WC	698.0	672.9	14.4	0.0	**-9.6**
			SC	111.7	60.8	8.9	0.0	**-0.9**
100	286	25	NC	**Infeasible**	0.2	1.1	–	**-7.9**
			WC	0.4	0.2	1.4	0.0	**-8.9**
			SC	0.7	0.4	1.5	0.0	**-4.0**
		33	NC	**Infeasible**	0.2	1.0	–	**-13.3**
			WC	1.0	0.6	2.0	0.0	**-7.6**
			SC	15.9	14.3	49.6	0.0	**-1.3**
		50	NC	4.3	1.8	4.0	0.0	**-6.2**
			WC	26.9	18.5	4.0	0.0	**-3.3**
			SC	199.8	103.5	201.3	0.0	**-3.5**
200	580	25	NC	0.3	0.3	4.0	**-7.8**	**-17.2**
			WC	0.5	0.4	4.0	**-9.1**	**-2.2**
			SC	3.2	2.7	3.9	0.0	**-3.8**
		50	NC	**Infeasible**	47.5	13.0	–	**-16.7**
			WC	52.9	52.9	7089.2	**-2.2**	**-7.7**
			SC	1265.4	1447.9	18.7	0.0	**-1.7**

Table 7.4: *Results for Letchford et al. [89]-Like instances with wide time windows (−: instances not solved in 7200 seconds)*

$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	CPU min-cost(s)	CPU min-time(s)	CPU Road Network(s)	gap min-cost(%)	gap min-time(%)
50	134	16	NC	2.0	1.3	15.0	**-12.4**	**-17.2**
			WC	31.4	16.3	2.8	0.0	**-11.3**
			SC	6.4	3.6	1.7	0.0	**-0.2**
		33	NC	**Infeasible**	2893.2	7125.2	–	**-11.4**
			WC	–	–	–	–	–
			SC	–	–	–	–	–
100	286	25	NC	1.4	1.9	28.2	**-5.3**	**-12.0**
			WC	1.1	0.8	2.4	0.0	**-4.6**
			SC	8.5	4.0	26.2	0.0	**-1.8**
		33	NC	9.5	1.3	2229.9	**-5.6**	**-9.0**
			WC	5.2	6.4	5.1	**-0.2**	**-3.5**
			SC	97.4	45.6	24.4	0.0	**-1.9**
		50	NC	22.2	10.2	6.1	0.0	**-6.1**
			WC	3361.3	2512.8	56.7	0.0	**-3.5**
			SC	–	–	1532.5	–	–
200	580	25	NC	0.9	15.5	7199.2	**-2.0**	**-15.8**
			WC	19.3	2.6	96.6	**-4.7**	**-1.5**
			SC	146.7	119.2	27.4	0.0	**-1.4**
		50	NC	1682.4	5150.0	–	–	–
			WC	4066.2	–	4364.3	**-3.2**	–
			SC	–	–	–	–	–

Table 7.5: *Results for real instances*

$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Ins	Construction (s)	CPU min-cost(s)	CPU Road Network(s)	gap min-cost(%)
5437	10181	5	1	12.8	0.02	7.0	**-10.4**
			2	17.8	0.03	8.2	**-11.8**
			3	17.2	0.10	24.4	**-7.6**
		10	1	28.3	0.06	18.3	**-7.1**
			2	43.0	0.05	12.7	**-6.9**
			3	34.9	0.05	40.9	**-2.1**
		25	1	129.3	0.09	97.9	**-0.6**
			2	114.6	0.09	56.8	**-1.4**
			3	164.3	0.08	35.8	**-4.3**

out of 36 solved instances compared to the min-time graph.

With the TDVRPTW$_{RN}$, which maintains the complete road-network information, solution costs are reduced for 10 instances compared to solutions obtained in the min-cost graph and for 36 instances compared to solutions obtained in the min-time graph. Gaps are up to 12.4% and 19.8%, respectively. Average gaps are 1.7% and 7.3%. Furthermore, the min-cost graph admits no feasible solution for 5 instances due to the loss in the graph construction of more costly but faster paths.

Another interesting observation is that increasing the correlation between costs and travel speeds reduces the improvements in solution costs obtained when conserving the complete road-network information. Especially, gaps are most of the times significantly larger for NC instances.

Table 7.5 summarizes results obtained for instances derived from the real road network. We observe that solving the TDVRPTW$_{RN}$ enables improving solution costs for all instances. The saving is 5.8% on average and reaches 11.8%. We also observe that the improvement in solution cost decreases when the number of customer nodes increases. The average improvement is 9.9% for instances with 5 customers and goes down to 2.1% for instances with 25 customers.

Finally, we notice that computing times are more important when addressing the TDVRPTW$_{RN}$ and increase significantly when the number of customers increases.

7.6 Conclusion

In this paper, we are interested in the Time-Dependent Vehicle Routing Problem with Time Windows and Road-Network Information (TDVRPTW$_{RN}$). The originality of this model is that time-dependent travel-time information is defined on road-network arcs. It goes against traditional modelings of time-dependent vehicle routing problems where a customer-based graph is first introduced and travel-times function are then directly defined on this graph arcs. With this work, we demonstrate that important savings can be obtained thanks to this new modeling.

In order to do so, several methodological contributions were needed. First, we propose two original dynamic programming algorithms to compute the travel-time and travel-distance functions in the min-cost and min-time graphs. Second, we adapt a branch-and-price scheme to the TDVRPTW$_{RN}$ to be able to derive optimal solutions while considering complete road-network information. The main specificity of this algorithm is to apply branch-and-price directly on the road-network graph.

This algorithm and the two branch-and-price algorithms for which vehicle routes are defined on customer-based graphs, show however some limits. Actually, the main purpose of our study was to point out the negative effects of the customer-based graphs against a complete consideration of road-network information. We did not seek for the development of state-of-the-art branch-and-price algorithms. For the TDVRPTW$_{RN}$, only instances with few customers (50 customers for simulated networks and 25 customers for large-scale real road networks) can be solved within the imparted computing time of two hours. The proposed branch-an-price framework could be enhanced in many ways. For example, the special structure of the road-network graph could be exploited much more in the pricing problem. Also, heuristic pricing methods might be designed. Introducing valid inequalities, in a branch-cut-and-price framework could also be very helpful. Another interesting research direction would consist in the development of heuristic solution methods, able to solve time-dependent vehicle routing problems directly on road networks.

Chapter 8

Conclusions and perspectives

Vehicle routing problems constitute one of the most studied classes of combinatorial optimization problems in the operations research literature. This is due to the large number of real-life applications where routing issues are involved.

Classically, these problems are tackled using the so-called customer-based graph, a complete graph representing the road network. In many situations, this modeling can have important consequences. One situation is when several attributes are defined on road segments. In this case, alternative paths with different compromises are not considered in the customer-based graph. This may have a negative impact on solution quality.

The first contribution of this thesis is to analyse works where the limits of the complete graph representation are evoked and that investigate vehicle routing problems with more information from the road network. We examine the different cases where the customer-based graph badly represents the road network and we highlight the alternative approaches proposed to handle these limits. The study shows the lack of contributions in this area and reveals many unexplored research directions.

In the literature, two alternative approaches are proposed to handle the limits of the customer-based graph representation when several attributes are defined on road segments. In the first approach, the road network is represented using a multigraph where an arc is introduced for each alternative path. In the second approach, the problem is solved directly on the road network.

A first part of this thesis focuses on the multigraph approach. We investigate the tractability of representing real road networks with multigraphs. We develop an exact solution method that computes the set of non-dominated paths linking a set of points of interest in a road network. The proposed algorithm is based on multi-destination multi-objective A* search strategy. Multigraphs for up to 500 points of interest are constructed within few seconds for large sized road networks. An analysis on the impact of the density of customers in the road network is conducted. Also, we study the impact of customers time windows on the number of feasible alternative paths and on the construction time.

In the second step, we investigate the impact of the multigraph representation on the quality of solution for a vehicle routing problem with two attributes: the VRPTW. We develop a branch-and-price algorithm that can handle the multigraph setting. An extensive computational study is carried out on modified instances from the literature and on instances derived from real road networks. Results show that, using the multigraph, solution costs are significantly reduced compared to the solution costs on the shortest-path-based graph (up to 14%) and compared to solution costs on the fastest-path-based graph (up to 54%). Moreover, we notice that using a multigraph increases only slightly the computing times compared to the customer-based graph. As another contribution, we develop a heuristic method that can efficiently handle the multigraph settings. The proposed method is based on an adaptive large neighborhood search able to explore different areas from the solution space. We embed into the algorithm an incremental data structure and a dynamic programming based procedure that allow to efficiently evaluate the neighborhood of a given solution in the presence of parallel arcs between nodes. Computational experiments show the competitiveness of the developed algorithm compared to the branch-and-price algorithm. Some improving solutions (compared to optimal solutions in simple graph representations) are obtained using the heuristic approach.

A second part of the thesis focus on the road network approach. We first investigate the relative efficiency of tackling the problem directly on the road network compared to the multigraph approach. For this aim, we develop a branch-and-price algorithm that works directly on the road network. An extensive computational study is then conducted to analyse the impact of characteristics of the VRPTW (elementary and non-elementary routes, customer time windows, capacity constraints, density of customers, etc.) on the performance of branch-and-price algorithms with the road network and the multigraph settings. Obtained results show that a multigraph-based branch-and-price scheme is more efficient than using the road network approach.

The second contribution of this part of the thesis is to point out the limits of the customer-based graph for a time dependent VRPTW. To do this, we first develop two algorithms to compute the customer-based graph of the road network. The first algorithm computes the shortest paths and the associated travel time functions from one to all other points of interest. The second algorithm permits to determine the fastest path between two points of interest and the associated distance for each possible departure time. Then, we adapt the road network based branch-and-price algorithm to handle the time dependent setting. Results show the potential benefits of the road network modeling and the attractive savings on solution costs achieved. It also comes out that optimal solutions on the road network are found for instances that are not feasible when using the shortest path based simple representation.

The primary interest of this thesis is to emphasize the limits of the standard modeling used to tackle a large class of vehicle routing problems. We focus on problems where several attributes are defined on road segments. However, there are many other situations where the

simple graph modeling is not suitable to efficiently vehicle routing problems. At the time of concluding this thesis, only a few papers investigate the impact of tackling problems using the simple graph representation. A natural extension of the work presented in this thesis would be to examine these situations and analyse the benefits of proceeding differently than using the standard modeling approach.

From a methodological point a view, future research would be interested in the development of efficient solution methods that can handle the road network setting. We recall that our objective in Chapters 6 and 7 was not achieve the best possible implementation of the branch-and-price algorithm for the road network based VRPTW. But, we sought for the development of a method that allows deriving conclusive results. The proposed algorithm can be enhanced in many ways. First, one would be interested in improving the branching rule. In the branching scheme proposed in Chapter 6, decisions mainly impact the set partitioning formulation and the lower bounds but do not affect the pricing problem. In addition, these decisions alone do not guarantee integrality of the solution. It would be interesting to design branching rules that allow obtaining an integer solution and have an impact on the structure of the pricing problem. A second possible enhancement is the design of specific dominance rules for the road network settings. In our implementation, we use the dominance rule introduced for the standard VRPTW. Using more suitable branching rules would improve significantly the efficiency of the pricing algorithm. Moreover, it would be interesting to implement a bounding test that eliminates non-dominated labels that, when extending in the best way, do not improve the current solution.

Another interesting research direction would be to design heuristic solution method that can handle the road network settings. Most heuristic approaches proposed for vehicle routing problems are based in local search moves that aim at relocating customer services to explore new solutions. Using the road network modelling, such operations are not easy to evaluate and to perform since customer nodes are not directly connected and several paths could exist between two points of interest. And many other computational challenges arise in this case.

Furthermore, the recent advances in information and communication technologies permit to obtain information on traffic conditions and travel times on real-time. Although the growing interest accorded to such information when tackling shortest path and route planning problems, there is a lack of contributions in this area for routing problems. It would be interesting to investigate the effect of considering such information for vehicle routing problems and to study the impact of both customer-based graph and road-network graph approaches on the solution quality for these problems. It would be also interesting to examine if it is sufficient to work with the customer-based graph representation for other categories of vehicle routing problems such as problems with possibility of diversion, problems with synchronization constraints, hazardous materials routing, etc. In such problems, several issues have to be considered, e.g., incident probability and population exposure for hazardous materials transportation, assignment of new service requests depending on actual situations

of the vehicles, etc. These issues can not be easily addressed using the traditional customer-based graph. Thus, it would be interesting to propose alternative modeling approaches for such problems.

Appendix A

Résumé en français

Problèmes de tournées de véhicules avec des informations du réseau routier

A.1 Introduction

Le transport présente une activité centrale dans la chaîne logistique dont le coût représente près de la moitié du coûts logistiques et peut atteindre jusqu'à 70% du coût total des biens dans certaines industries [33]. Selon la Commission européenne, l'industrie du transport a généré jusqu'à 7% du produit intérieur brut (PIB) de l'Union Européenne (UE) en 2009 [101]. Aux États-Unis, les charges totales de la logistique ont atteint 10% du PIB en 2000 et les coûts de transport représentaient plus que la moitié de cette contribution [69]. De plus, les activités de transport ont un impact environnemental très important. Les consommations en carburant et en énergies dues au transport représentent, respectivement, 60% et 25% des de la consommation globale [116]. L'Agence Européenne pour l'Environnement indique qu'*en* 2013 *le secteur du transport a contribué à environs le quart (*$24,4\%$*) des émissions totales de gaz à effet de serre de l'UE* [46].

Par conséquent, il est trés important de concevoir un système de transport afin d'accroître la compétitivité d'une entreprise et de diminuer son impact environnemental. Cette tâche donne lieu à un problème bien connu de la littérature : le problème tournées de véhicules (VRPs). Ce problème peut être rencontré dans plusieurs applications réelles qui impliquent des activités de transport de passagers ou de marchandises (e.g. transport de personnes handicapées, tournées des autobus scolaires, livraison du courrier, collecte des déchets, ramassage et livraison des marchandises,...).

Le problème de tournées de véhicules consiste à calculer l'ensemble des routes à coût minimal qui permettent de servir un ensemble de clients géographiquement dispersés. Les problèmes de tournées de véhicules ont fait l'objet de plusieurs travaux de recherche depuis maintenant plus de 50 ans. Depuis l'introduction du VRP par Dantzig et Ramser [31] en 1959, des milliers d'articles et de livres ont été consacrés à l'étude des problèmes d'optimisation

dans lesquels des problèmes de routage sont impliqués. Eksioglu et al. [51] énumérent environ 1500 publications indexées qui portent sur des problèmes de tournées de véhicules.

Classiquement, les problèmes de tournées de véhicules sont définis sur des réseaux routiers où les demandes de services (ramassages ou livraisons) sont associées à des points spécifiques. Donc, la qualité de la solution dépend fortement de la qualité de la réprésentation du réseau routier. La plupart des approches rencontrées dans la littérature s'appuient sur un graphe complet où un nœud est introduit pour tout point d'intérêt du réseau routier (typiquement les clients et le dépôt). Cette modélisation est, implicitement, basée sur l'hypothèse que le meilleur chemin entre toute paire de points du réseau routier est bien défini. Cependant, cette hypothèse n'est pas toujours valide dans de nombreuses situations. Souvent, plus d'informations sont nécessaires pour modéliser et résoudre correctement le problème.

Une première situation est celle où plusieurs attributs sont définis sur des segments de route. Dans ce cas, la définition du meilleur chemin entre deux points du réseau routier implique un probème d'optimisation multi-objectif. La solution de ce problème consiste en l'ensemble des chemins non dominés. En considérant un seul chemin, de bonnes solutions, potentiellement optimales, pourraient être éliminées de l'espace des solutions. Ce qui résulte en une surestimation du coût de la solution.

A notre connaissance, Garaix et al. [61] ont été les premiers à souligner que, lorsque plusieurs attributs sont associés aux segments de route, modéliser le problème avec un graphe complet peut conduire à une surestimation du coût de la solution. Ils étudient un problème de transport à la demande avec l'objectif de développer un système de transport à la demande réel pour une zone rurale. Ils proposent d'aborder le problème en utilisant un multigraphe afin de gérer efficacement le compromis coût-temps. Ils poursuivent deux objectifs principaux: évaluer la faisabilité de cette approche et démontrer ses avantages par rapport à l'utilisation du graphe complet traditionnel. Plusieurs auteurs se sont appuyés sur ces résultats et ont examiné l'intérêt de la modélisation par multigraphe pour différents problèmes: Lai et al. [85] pour un VRP avec une flotte mixte et des contraintes de temps, Ben Ticha et al. [8, 7] pour le problème de tournés de véhicules avec des fenêtres de temps (VRPTW), Huang et al. [71] pour un VRP avec des temps de parcours qui varient au cours du temps. Toutes ces études se sont particulièrement intéressées à l'évaluation des gains qui peuvent être obtenus avec le multigraphe. Des analyses expérimentales approfondies utilisant différentes classes d'instances sont proposées. Elles montrent les gains importants qui peuvent être obtenus en utilisant le multigraphe.

Une deuxième approche est aussi proposée afin de conserver l'ensemble de l'espace de solutions et retrouver la solution optimale. Cette approche consiste à résoudre le probléme directement sur un graphe simulant le réseau routier. Dans ce graphe, les arcs correspondent aux segments de route et les un nœuds aux extrémités de ces segments. Un article clé à cet égard est proposé par Letchford et al. [89]. Ils se sont intéressés à l'approche proposée

par Garaix et al. [61]. Ils étaient guidés par l'idée qu'il serait, peut-être plus efficace de résoudre le problème sur le graphe du réseau routier au lieu d'introduire la représentation par multigraphe. En particulier, Letchford et al. [89] soulignent que, dans le pire cas, la construction et le stockage du multigraph pourraient être exponentiels dans le temps et dans l'espace (ce qui était contredit dans [8, 9]). Ils identifient l'impact de considérer le graphe du réseau routier sur l'algorithme du branch-and-price mais seul un algorithme de génération de colonnes est développé. Le schéma de branchement est laissé pour de futures recherches. Les résultats numériques démontrent l'intérêt de leur approche lors du calcul de la relaxation linéaire continue du problème, par rapport à une procédure de génération de colonnes équivalente appliquée au multigraphe.

La plupart des travaux de recherche qui portent sur des VRPs supposent que tous les attributs dans le réseau routier sont bien définis a priori. Ces attributs sont donnés en entrée ou calculés en utilisant d'autres informations données, par exemple, les temps de parcours sont calculés en utilisant les distance et les vitesses moyennes. Par conséquence, le meilleur chemin entre deux paires de nœuds peut être facilement défini quand un seul attribut est considéré sur le réseau routier. Cependant, il existe certaines situations où calculer le meilleur chemin pour un attribut donné entre deux points d'intérêt n'est pas si simple. Un exemple simple est celui des temps de parcours dépendant du temps. En théorie, il est facile de calculer le chemin le plus court en temps pour chaque heure de départ possible. Ce qui permet ensuite de construire la représentation par graphe complet avec des fonctions de temps de parcours associées à chaque arc. Il faut juste être conscient que le chemin du réseau routier associé à un arc dans ce graphe peut changer selon le moment de la journée. Même en pratique cela peut être plus compliqué (en raison de problèmes de mémoire ou de précision liés à la modélisation des données de temps de déplacement, comme discuté dans Eglese et al. [47]).

D'autres complications surviennent quand la fonction objectif est dépendante du temps. C'est le cas, par exemple, quand l'objectif est de déterminer l'ensemble de routes qui minimise la consommation de carburant ou la minimisation des émissions de carbone, avec des temps de parcours dépendant du temps [134]. Dans ce cas, le problème du chemin de coût minimum entre deux points devient non trivial si le principe d'optimalité de Bellman n'est pas valide, c'est-à-dire s'il est préférable d'arriver à un nœud intermédiaire avec un coût plus élevé (mais plus tôt) afin d'éviter la congestion et économiser des coûts futurs. Une difficulté supplémentaire est aussi rencontrée si la consommation de carburant est modélisée en fonction de la charge du véhicule. Dans ces situations, la construction du graphe complet devient de moins en moins viable.

Encore plus compliqué sont les situations où le décideur a la possibilité de prendre des décisions qui pourraient influencer l'itinéraire à suivre entre deux points d'intérêts. Un premier exemple est quand le décideur a la possibilité de contrôler la vitesse de parcours afin d'optimiser la consommation en carburant [109]. Un autre exemple est quand le décideur

doit gérer les pauses des conducteurs afin d'assurer le respect des règles des horaires de travail (voir [19]). Dans ce cas, il est difficile, voire impossible, de définir a priori le meilleur chemin entre deux points d'intérêt et donc de construire la représentation par graphe complet du réseau routier.

Une troisième situation où le graphe complet peut mal représenter le réseau routier est quand ce dernier présente une structure complexe. En effet, le transport de marchandises constitue une activité importante dans les zones urbaines. La plupart des réseaux routiers ont une structure complexe et des informations additionnelles devraient être considérées en traitant de problèmes de tournées de véhicules.

Récemment, de nombreuses études dans la littérature du VRP s'intéressent à l'optimisation des activités de transport et de distribution dans les centres urbains, ce qu'on appelle la logistique urbaine (*city logistics*). Dans la plupart de ces études, le problème est défini sur un graphe complet et trois types de décisions sont principalement traités: les décisions d'affectation et de séquencement qui définissent l'ordre de visite du client pour chaque véhicule et les décisions d'ordonnancement qui déterminent l'horaire de visite. Implicitement, il est supposé que le chemin choisi entre deux clients ne dépend pas de la séquence. En d'autres termes, on suppose une totale indépendance dans les décisions de sélection de chemin. Cependant, cette hypothèse n'est pas toujours pertinente dans les réseaux routiers complexes comme ceux trouvés dans les zones urbaines. En effet, la façon dont un client est atteint pourrait influencer la façon avec laquelle il sera quitté. Dans ce contexte, une étude est proposée par Lang et al. [86]. Ils examinent l'impact de considérer des points de stationnement alternatifs auxquels le véhicule peut s'arrêter pour servir le client. Lang et al. [86] présentent un cas d'étude avec des données du réseau routier de Beijing (Chine). Les résultats obtenus montrent l'impact intéressant de leur approche sur la qualité de la solution. Un second type de dépendance existe lorsque l'accès à certaines zones du réseau est soumises à des frais. Selon que ces frais sont payés ou non, le meilleur chemin entre deux points du réseau routier ne sera pas le même. Reinhardt et al. [113] évoquent une telle situation. Ils se sont intéressés au problème de tournées de véhicules où des coûts supplémentaires doivent être payés pour avoir l'accès à chaque sous-ensemble de connexions dans le réseau. Ces coûts peuvent correspondre au paiement pour les routes à péage, les liaisons par ferry, l'investissement dans de nouvelles installations ou des certifications pour accéder aux zones de guerre ou aux zones de troubles. Dans ce cas, le coût du meilleur chemin entre deux clients dépend des décisions d'achat de l'accès à des ensembles de connections dans le réseau routier. Ainsi, le problème ne peut être modélisé correctement avec une représentation par graphe complet. Reinhardt et al. [113] expliquent que le problème devrait être modélisé en utilisant un multigraphe afin de différencier les arêtes reliant la même paire de nœuds mais n'appartenant pas aux mêmes sous-ensembles.

Enfin, d'un point de vue méthodologique, transformer le problème de tournées de véhicules sur le réseau routier en un problème de tournées de véhicule standard pourrait avoir un im-

pact négatif sur les performances de calcul. Bien que le graphe complet réduise le nombre de nœuds par rapport à une représentation graphique du réseau routier d'origine, il peut augmenter considérablement l'ensemble des arcs. Ainsi, il empêche d'exploiter certaines propriétés, telles que la planéité du graphe et le faible degré des nœuds.

Fleischmann [56] fut le premier à explorer ce problème. Il affirme que les nœuds dans les réseaux routiers ont typiquement de petits degrés et que le nombre d'arcs dans un graphe de réseau routier peut être largement inférieur à celui dans un graphe complet basé sur les nœuds clients. Ainsi, quand le problème est traité avec un modèle de flux, transformer le réseau routier en un graphe complet peut augmenter de manière critique le nombre de variables. L'auteur considère d'abord le problème d'optimisation de tournée d'un seul véhicule dans un graphe de réseau routier, appelé problème de voyageur de commerce de Steiner ou aussi *Steiner Traveling Salesman Problem (STSP)*. Il propose une méthode de solution basée sur un schéma de plans de coupe. Dans le même article, l'auteur propose également d'étendre la méthode de résolution au VRP et souligne les principales difficultés qui se poseraient dans ce cas.

Plus récemment, Letchford et al. [90] ont étudié plus en profondeur l'impact de la prise en compte du réseau routier lors de la modélisation du TSP. Ils proposent différentes formulations mathématiques pour le STSP avec un nombre polynomial de variables et de contraintes. Les résultats expérimentaux montrent que les instances avec 200 nœuds sont résolues en utilisant un solveur de programmation mathématique standard.

Dans la littérature, le nombre d'articles traitant de ces questions est très limité. Les quelques études proposées à ce sujet n'examinent pas l'impact de la modélisation du problème de tournées de véhicules par un graphe complet sur la qualité de la solution et se sont, pour la plupart, intéressées à la résolution de problèmes spécifiques.

L'un des objectifs de cette thèse est de mettre en évidence les limites de la modélisation des problèmes de tournées de véhicules par graphe complet et de confirmer la nécessité de nouveaux modèles et approches permettant de considérer plus d'informations sur le réseau routier afin de résoudre correctement ces problèmes. Pour ce faire, nous nous concentrons sur les problèmes de tournées de véhicules où plusieurs attributs sont définis sur les segments de route. Comme nous l'avons déjà mentionné, ces problèmes sont généralement traités via un graphe complet basé sur les nœuds clients, où un arc est supposé représenter le meilleur chemin entre les points d'intérêt. Cependant, il est peu probable que le meilleur chemin soit le même pour tous les attributs. Les chemins avec des compromis différents ne sont donc pas pris en compte dans ce graphe. Pour gérer ce problème, deux approches sont proposées dans la littérature. La première approche consiste à représenter le réseau routier à l'aide d'un multigraphe où un arc est introduit pour chaque chemin alternatif. La deuxième approche consiste à résoudre le problème de tournées de véhicules directement sur un graphe qui imite le réseau routier d'origine, que nous appelons graphe route-réseau.

Dans cette thèse, nous commençons par explorer la représentation du réseau routier par multigraphe. Nous étudions la construction de cette représentation et nous proposons une méthode de résolution efficace qui permet de calculer l'ensemble des chemins alternatifs pour chaque paire de points d'intérêt dans le réseau routier. Ensuite, nous étudions la traçabilité de cette modélisation pour les problèmes de tournées de véhicules et nous analysons en profondeur l'impact de cette approche sur la qualité de la solution. Dans une seconde étape, nous nous concentrons sur le développement d'une méthode de solution efficace qui permet de bien gérer la modélisation par multigraphe. Dans une troisième étape, nous explorons l'approche basée sur le réseau routier. Nous proposons une comparaison complète des deux approches. Nous présentons une étude expérimentale approfondie basée sur des problèmes de référence ainsi que des instances générées en utilisant des données réelles. Dans une dernière étape, nous nous intéressons aux problèmes où les temps de parcours varient au cours de la journée, ce que l'on appelle les problèmes de tournées des véhicules dépendant du temps (TDVRP). Nous expliquons que, dans de tels contextes, il est très difficile voire impossible de s'attaquer au problème en utilisant la représentation par multigraphe. Pour cette raison, nous développons un algorithme de branch-and-price pour résoudre le problème directement sur le graphe du réseau routier. Enfin, nous analysons l'impact de l'approche proposée sur la qualité de la solution pour les problèmes de routage des véhicules en fonction du temps.

A.2 Construction de la représentation par multigraphe du réseau routier

Dans la modélisation du VRP par multigraphe, un nœud est introduit pour chaque point d'intérêt (typiquement les clients et le dépôt) et un arc est introduit pour chaque chemin efficace entre deux points d'intérêt dans le réseau routier. La construction d'une telle représentation consiste, donc, à calculer l'ensemble de chemins efficaces reliant chaque paire de points d'intérêt ce qui implique la résolution d'un problème de plus court chemin multi-objectif.

Dans la première partie de la thèse, nous étudions la construction de la représentation multigraphe pour les problèmes de tournées de véhicules où chaque segment de route est défini par une distance et un temps de parcours. Nous proposons une méthode de résolution qui calcule l'ensemble des chemins les plus courts bi-objectifs d'un à tous les autres points d'intérêt dans un réseau routier. Dans la littérature, presque toutes les approches proposées visent à calculer soit des chemins d'un point à un autre (*one-to-one*), soit des chemins d'un point source à tous les autres points du réseau (*one-to-all*). Bien que le cas multi-destination puisse être abordés en utilisant un algorithme pour l'un des deux autres cas (en appliquant un algorithme one-to-one pour chaque nœud de destination ou en considérant uniquement les chemins arrivant aux nœuds requis dans une solution one-to-all) , des propriétés intéres-

santes apparaissent dans ce cas et pourraient être exploitées pour résoudre le problème plus efficacement. La méthode de résolution que nous proposons permet de traiter le cas multi-destination d'une manière efficace.

Nous définissons le réseau routier comme un graphe orienté $G = (V, A)$ où V est un ensemble de n nœuds et A est un ensemble de m arcs. Chaque arc $(i, j) \in A$ est défini par deux valeurs non négatives (d_{ij}, t_{ij}) représentant les coûts pour les deux objectifs considérés. Dans notre application, les coûts d_{ij} et t_{ij} correspondent respectivement à la distance et au temps de parcours. Soit $v_0 \in V$ le nœud source et $C = \{v_1, v_2, \ldots, v_{nc}\} \subset V$ le sous-ensemble de nœuds requis, c'est-à-dire l'ensemble des nc nœuds de destination.

Dans la littérature, l'algorithme le plus utilisé pour résoudre un tel problème est basé sur une procédure de labelling ou un chemin entre le nœud source v_0 et un nœud i est représenté par un label $L = (i, d(L), t(L))$ ou $d(L)$ et $t(L)$ correspondent, respectivement, à la distance totale et le temps de parcours total associés au chemin représenté par L. Dans cet algorithme (voir **Algorithme A.1**), tous les labels sont maintenus dans une liste $\mathcal{L}$ et sont triés suivant un ordre lexicographique. A chaque itération, un label est sélectionné et est étendue vers tous les arcs sortants. A chaque fois qu'un nouveau label est obtenu, un contrôle de dominance est effectué et seuls les labels non-dominés sont maintenus dans $\mathcal{L}$. L'algorithme se termine quand tous les labels sont traités, c'est-à-dire $\mathcal{L} = \emptyset$.

Algorithm A.1 Algorithme de labelling

1: $\mathcal{L} \leftarrow \{(v_0, 0, 0)\}$
2: **repeat**
3: Select $L = (i, d, t) \in \mathcal{L}$
4: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{L\}$
5: **for all** j successor of i **do**
6: $L' = (j, d + d_{ij}, t + t_{ij})$
7: InsertWithDominance($L', \mathcal{L}$); $L_1 \prec L_2 \Leftrightarrow i_1 = i_2$ and $t_1 \leq t_2$ and $d_1 \leq d_2$
8: **end for**
9: **until** $\mathcal{L} = \emptyset$

Dans notre approche, nous proposons de guider la recherche en utilisant une stratégie A*. Fondamentalement, l'algorithme A* construit le chemin le plus court en étendant d'abord les labels qui semblent conduire plus rapidement à la meilleure solution [68]. Ces labels sont sélectionnés sur la base d'une estimation du coût restant pour aller au nœud destination v. Dans notre implémentation, nous proposons de sélectionner, à chaque itération, parmi tous les labels, celui qui peut conduire à un chemin non-dominé avec la distance la plus courte (ligne 3 dans l'algorithme A.1). En d'autres termes, nous sélectionnons à chaque itération le label $L = (u, d(L), t(L))$ qui minimise $d(L) + d^{inf}(u, v)$ où $d^{inf}(u, v)$ représente la plus courte distance pour aller de u à v. De cette manière, les chemins arrivant au nœud de destination sont générés dans un ordre de distance croissant. Ainsi, la recherche peut être terminée une

fois le label sélectionné est tel que $d(L)+d^{inf}(u,v) > d^{sup}(v_0,v)$ avec $d^{sup}(v_0,v)$ est la distance associée au chemin le plus court en temps pour aller de v_0 à v ($t^{inf}(v_0,v)$ dénote le temps de parcours associé). Nous avons montré qu'une fois que ce critère d'arrêt est satisfait, il est garanti que tous les chemins non dominés sur le nœud destination v ont été trouvés. Notez que le calcul des valeurs $d^{inf}(u,v)$ et $d^{sup}(v_0,v)$ peut être fait en pré-traitement.

Pour gérer le cas multi-destination, nous proposons d'adapter le schéma de recherche de telle sorte que la procédure de sélection prenne en compte toutes les destinations finales pour chaque label. La fonction de sélection vise donc à définir le label susceptible de conduire le plus rapidement à un chemin non dominé à un certain nœud destination. Pour ce faire, on sélectionne le label $L = (u, d(L), t(L))$ qui minimise :

$$\min_{1\leq i\leq nc} (d(L) + d^{inf}(u,v_i) - d^{inf}(s,v_i))$$

Nous notons par $K(L)$ cette valeur et l'appelons la clé du label. Il indique le détour minimum en distance du sous-chemin associé étant donné qu'il devra atteindre l'une des destinations en C.

En utilisant cette procédure de recherche, les chemins atteignant chaque nœud destination sont générés dans un ordre croissant de détour en distance. L'algorithme devrait, donc, terminer une fois que les labels $L_i = (v_i, d^{sup}(v_0,v_i), t^{inf}(v_0,v_i))$ ont été générées pour chaque nœud de destination $v_i \in \{v_1, v_2, \ldots, v_{nc}\}$. Par conséquent, le critère d'arrêt est satisfait quand la clé du label sélectionné est supérieure à $\max_{1\leq i\leq nc} K_i^{sup}$ avec $K_i^{sup} = d^{sup}(v_0,v_i) - d^{inf}(v_0,v_i)$.

Pour calculer la clé d'un label $L = (u, d(L), t(L))$, les distances $d^{inf}(u,v_i)$ et $d^{inf}(v0,vi)$ associées, respectivement, aux chemins les plus courts en distance de u à v_i et de v_0 à v_i sont nécessaires. De plus, les distances $d^{sup}(v_0,v_i)$ (utilisées pour calculer K_i^{sup}) associées aux plus courts chemins en temps de v_0 à v_i sont nécessaires. Ces distances sont déterminées en pré-traitement comme suit:

- En utilisant l'algorithme de Dijkstra , nous calculons les chemins les plus courts en distance et en temps du nœud v_0 vers tous les nœuds $u \in V$. Quatre tables sont construites: $d^{inf}(v_0,u)$ et $t^{sup}(v_0,u)$ indiquant les distances et les temps associés aux plus courts chemins en distance et, $d^{sup}(v_0,v)$ et $t^{inf}(v0,v)$ indiquant les distances et les temps associés aux plus courts chemins en temps.

- En utilisant un algorithme de Dijkstra rétrograde à partir de tous les nœuds $v_1, \ldots, v_{nc}$, nous calculons les chemins les plus courts en distance et en temps de tous les nœuds $u \in V$ vers les nœuds de destination $v_i \in \{v_1, \ldots, v_{nc}\}$. Quatre séries de tables sont obtenues: $d^{inf}(v,v_i)$ et $t^{sup}(v,v_i)$ indiquant les distances et les temps associés aux plus courts chemins en distance et, $d^{sup}(v,v_i)$ et $t^{inf}(v,v_i)$ indiquant les distances et temps associés aux plus courts chemins en temps.

Afin d'évaluer la performance de notre algorithme, nous comparons les résultats obtenus à ceux obtenus en utilisant des algorithmes de la littérature: l'algorithme de labelling de base présenté précédemment (LSET) et un algorithme à correction de label (LCOR). Dans LCOR, les labels arrivant à chaque nœud sont comparés entre eux et seuls les non-dominés sont maintenus dans une liste à part. A chaque itération, un label arrivant à un nœud i est sélectionné et est étendu en utilisant les arcs (i, j).

Puisque nous nous intéressons au calcul de chemins pour des problèmes de transport, nos expérimentations sont basées sur des instances générées à partir de trois réseaux routiers réels: deux réseaux routiers de la ville d'Aix-en-Provence (France) et un réseau routier de la ville de Washington DC. (États Unis). Pour chaque réseau routier, 5 instances sont générées avec $nc \in \{25, 50, 100, 200, 500\}$ nœuds client.

Les résultats obtenus montrent l'efficacité de notre algorithme. Comparé aux algorithmes de labelling de base, notre algorithme surpasse l'algorithme LSET pour toutes les instances et améliore les temps de calcul avec l'algorithme LCOR pour la plupart des instances avec un grand nombre de nœuds dans les réseaux routiers.

A.3 Analyse empirique pour le VRPTW avec une représentation par un multigraphe du réseau routier

Dans ce chapitre, nous examinons en profondeur l'impact de la représentation par multigraphe sur la qualité de la solution pour les problèmes de tournées de véhicules dans les réseaux routiers. Pour ce faire, nous considérons le problème de tournées de véhicules avec des fenêtres de temps (VRPTW) comme un problème pilote et nous développons un algorithme de branch-and-price pour le résoudre.

Le VRPTW dans un multigraphe est défini comme suit. Soit $G^{MG} = (V^{MG}, A^{MG})$ un multigraphe orienté induit par le réseau routier $G = (V, A)$. $V^{MG} = \{0, 1, \ldots, nc\}$ est l'ensemble des nœuds (points d'intérêt dans G) où 0 représente le dépôt et $C = \{1, \ldots, nc\}$ est l'ensemble des nœuds clients. $A^{MG} = \bigcup_{i,j \in V^{mG}} A_{ij}^{MG}$ est l'ensemble d'arcs avec $A_{ij}^{MG} = \{(i, j)^p; p = 1, \ldots, |A_{ij}^{MG}|\}$ représente l'ensemble de chemins alternatifs pour aller de i à j. Chaque arc $(i, j)^p$ est défini par un coût $c_{(i,j)^p}$ et un temps de parcours $t_{(i,j)^p}$ qui représentent le coût et le temps de parcours pour aller de i à j à travers le chemin associé et indexé par p dans le réseau routier. A chaque nœud client sont associés une demande d_i, une fenêtre de temps $[e_i, l_i]$ et un temps de service s_i. Pour servir l'ensemble des clients, on utilise une flotte $mathcalK$ de véhicules homogènes de capacité Q. L'objectif est de définir un ensemble de routes qui permet de servir l'ensemble de clients et qui minimisent le coût total.

Pour résoudre ce problème, nous avons développé un algorithme de branch-and-price qui tient compte de la modélisation par multigraphe. La motivation pour utiliser une technique de branch-and-price est que cette approche de résolution a très bien fonctionné pour de nombreux problèmes de transport et est devenue l'une des méthodes exactes les plus efficaces pour résoudre des problèmes de tournées de véhicules.

L'algorithme de branch-and-price est basé sur la décomposition de Dantzig-Wolfe [32]. Cette décomposition donne lieu à un problème en nombres entiers, dit problème maître, qui présente une relaxation linéaire plus serrée que la formulation compacte par variables de flux. Par conséquent, la formulation utilisant le problème maître est plus appropriée pour un schéma de branchement. Cependant, cette formulation implique un grand nombre de variables. Pour gérer ces variables, une procédure de génération de colonne est intégrée dans l'algorithme de branch-and-bound. Le sous-problème de génération de colonnes, appelé problème de pricing, vise à trouver un ensemble de colonnes réalisables qui seront ajoutées au problème principal [41, 39, 41].

Soit Ω l'ensemble des routes de véhicules réalisables, c'est-à-dire l'ensemble des chemins dans le multigraphe qui partent du dépôt, visitent un sous-ensemble de clients, en respectant les contraintes de fenêtres de temps et les contraintes de capacité, et retournent au dépôt. Ainsi le VRPTW dans le multigraphe peut être formulé comme un problème de couverture par ensembles (appelé problème maître) où une variable est associée à chaque route et indiquant si la route est sélectionnée ou pas dans la solution. Due au nombre exponentiel de routes dans Ω, le problème maître ne peut être résolu avec un algorithme de branch-and-bound standard. Un algorithme de branch-and-price est utilisé avec une technique de génération de colonnes.

Dans une procédure de génération de colonnes, un problème maître restreint à un sous ensemble de route $\Omega_1 \subset \Omega$, qu'on note par $MP(\Omega_1)$, est résolu à chaque itération. Le sous-ensemble Ω_1 est enrichi itérativement avec les routes réalisables générées en utilisant un algorithme de pricing. Cet algorithme vise à générer de nouvelles routes permettant de visiter les clients d'une manière plus efficace, c'est-à-dire, des routes avec des coûts réduits négatifs.

Comme pour le VRPTW standard, le problème de pricing pour la variante multigraphe peut être réduit à un problème de plus court chemin élémentaire avec des contraintes de ressources (ESPPRC). Ici, l'ESPPRC consiste à trouver dans un multigraphe l'ensemble des chemins élémentaires qui partent et rentrent au dépôt, et qui satisfont les contraintes de temps et de capacité. Pour résoudre ce problème, nous adaptons l'algorithme de programmation dynamique décrit dans [53] au cas multigraph de l'ESPPRC. Fondamentalement, l'algorithme est une extension de l'algorithme de Bellman-Ford et consiste à associer à chaque chemin partiel un label et à étendre ces labels.

Avec la représentation en multigraphe, plus d'un label peuvent être obtenus lors de

l'extension d'un chemin partiel à un client j. Un nouveau chemin est obtenu pour chaque arc connectant le dernier nœud i du label courant à j tant que les contraintes de ressources le permettent. Nous utilisons des règles de dominance basées sur la consommation de ressources pour éliminer les labels dominés par d'autres labels arrivant au même nœud. A la fin, l'algorithme donne les meilleurs chemins possibles.

Une autre composante de l'algorithme de branch-and-price est le schéma de branchement qui consiste à étendre de manière itérative l'arbre de recherche binaire en ajoutant des contraintes impliquées par les règles de branchement. Dans le contexte des problèmes de tournées de véhicules, la règle de branchement standard consiste à sélectionner un arc (i, j) avec un flux fractionnaire $0 < \phi_{ij} < 1$ puis à dériver deux branches où dans la première branche l'arc (i, j) doit être traversé dans la solution, et il est interdit dans la deuxième branche. Pour la modélisation par multigraphe, nous utilisons une règle de branchement similaire qui fonctionne comme suit:

- Sélectionner un arc $(i, j)^p$ avec un flux fractionnaire $0 < \phi_{(i,j)^p} < 1$
- dériver deux branches
  - Dans la première branche, nous forçons l'utilisation de l'arc $(i, j)^p$ dans la solution. Pour ce faire, nous enlevons de Ω_1 toutes les routes utilisant les arcs $(i, j)^q$; $q \neq p$, (arcs parallèles à $(i, j)^p$), les arcs $(i, k)^l$; $k \neq j$, $l = 1, \ldots, |A_{ik}^{MG}|$ et les arcs $(k, j)^l$; $k \neq i$, $l = 1, \ldots, |A_{kj}^{MG}|$. Nous supprimons également tous ces arcs du multigraphe considéré dans le problème de pricing.
  - Dans la deuxième branche, nous interdisons l'utilisation de l'arc $(i, j)^p$ dans la solution. Pour ce faire, nous éliminons de Ω_1 toutes les routes utilisant l'arc $(i, j)^p$ et supprimons cet arc du multigraphe considéré dans la phase de pricing.

Afin de tirer des conclusions globales, nous avons utilisé 4 classes d'instances: (1) les instances de Solomon [126]; (2) des instances réalistes de Letchford et al. [89]; (3) des instances construites comme dans [89]; (4) des instances construites à partir des données réelles de deux réseaux routiers de la ville d'Aix-en-Provence (France). Aussi afin d'avoir une comparaison juste et complète, nous comparons les solutions obtenues dans le multigraphe à celles obtenues dans deux graphes complets: dans le premier graphe, qu'on appelle *min-cost graphe*, un arc représente le plus court chemin en coût entre deux points d'intérêt et dans le deuxième graphe, qu'on appelle *min-time graphe*, un arc représente le plus court chemin en temps entre deux points d'intérêt.

Les résultats obtenus montrent qu'en utilisant le multigraphe, le coût de la solution est amélioré pour 84 instances (parmi 177 instances testées) dans le min-cost graphe et est amélioré pour 136 instances dans le min-time graphe. Le coût de la solution est réduit jusqu'à 15% dans le min-cost graphe et jusqu'à 54% dans le min-time graphe. Les gains moyens en

coût sont de 2% et 7.5% comparé aux solutions obtenues respectivement dans le min-cost graphe et le min-time graphe.

Pour les instances réelles, en utilisant la modélisation par multigraphe toutes les solutions optimales obtenues dans les deux graphes complets sont améliorées. Le coût de la solution est réduit jusqu'à environ 10% dans les deux graphes. Les gains moyens sont de 3.3% et 6.6%, respectivement, dans le min-cost graphe et le min-time graphe.

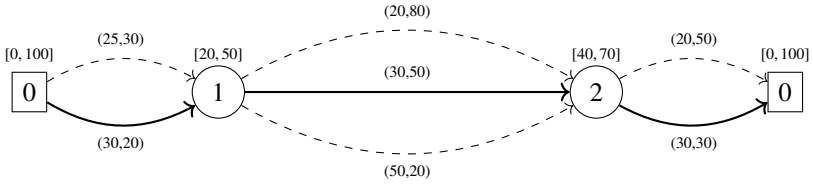
A.4 Recherche adaptative à voisinage large pour le VRPTW dans le multigraphe

Dans ce chapitre, nous nous sommes intéressé au développement d'une méthode de résolution efficace pour le VRPTW dans le multigraphe. Nous proposons une méthode heuristique basée sur une recherche adaptative à voisinage large.

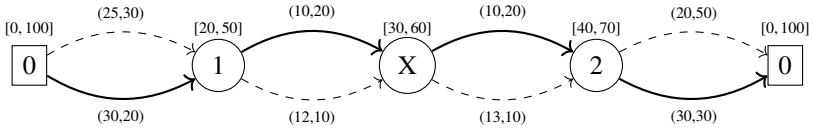
Avec la modélisation par multigraphe, certains problèmes de calcul se posent. En particulier, il a été montré que le calcul de la route optimale pour une séquence donnée de clients sur le multigraph implique un problème NP-difficile, appelé problème de sélection d'arc pour une séquence fixée (FSASP) [61]. Or, la plupart des opérations de recherche locale sont basées sur la suppression ou l'insertion de clients et nécessitent la ré-optimisation de la sélection d'arcs à utiliser pour connecter la nouvelle séquence de nœuds clients. Ainsi, même les mouvements de recherche locale simples deviennent difficiles à évaluer dans le multigraphe. Pour illustrer ceci, considérons l'exemple présenté dans la Figue A.1.

Dans la figure A.1a, est donnée une route initiale définie par la séquence de nœuds $\{0, 1, 2, 0\}$. Une fenêtre de temps est associée à chaque nœud. Des arcs parallèles entre chaque paire de nœuds définissent les chemins alternatifs entre les deux clients associés. Chaque arc est défini par un coût et un temps de déplacement et sont donnés entre parenthèses, dans cet ordre (coût, temps). Les arcs sélectionnés dans la route sont représentés par des lignes épaisses. Supposons qu'on cherche à évaluer l'insertion du client X entre les nœuds clients 1 et 2. Typiquement, cela se fait en sélectionnant les meilleurs arcs liant 1 avec X et X avec 2 et assurant la faisabilité de la nouvelle route. Dans ce cas, la route obtenue est fournie dans la Figure A.1b et a un coût total égal à 80. Cependant, si nous recalculons toute la séquence optimale d'arcs à utiliser pour lier la nouvelle séquence de nœuds $\{0, 1, X, 2, 0\}$, une nouvelle route (voir Figure A.1c) est obtenue avec un meilleur coût total égal à 60.

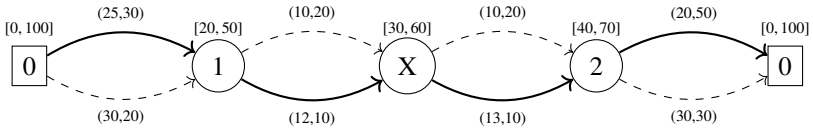
Pour gérer efficacement la sélection d'arcs au cours de la recherche, nous intégrons dans notre heuristique un algorithme basé sur une structure de donnée incrémentale et de la programmation dynamique. Étant donné une solution initiale, cet algorithme commence par



(a) *Route initiale*



(b) *Route après une insertion d'un client et sans le ré-optimisation de la sélection d'arcs*



(c) *Route optimale après l'insertion du client*

Figure A.1: *Illustration de l'insertion d'un client*

calculer pour chaque nœud dans chaque route l'ensemble des labels en *forward*, c'est-à-dire en partant du dépôt, et l'ensemble de labels en *backward*, c'est-à-dire arrivant au dépôt en partant du nœud. Ces labels sont maintenus tout au long de la procédure de recherche et mis-à-jour à chaque fois que la solution courante est modifiée.

Notre heuristique est basée sur une variante de la recherche à voisinage large (LNS), dite recherche adaptative à voisinage large (ALNS). L'idée principale de l'heuristique LNS est de réorganiser, à chaque itération, une grande partie d'une solution en utilisant une combinaison de procédure de destruction et de réparation. La procédure de destruction consiste à supprimer un nombre prédéfini de clients. La procédure de réparation vise à réinsérer les clients supprimés afin de créer une nouvelle solution réalisable. Dans un schéma de recherche locale, l'algorithme examine le voisinage de la meilleure solution trouvée à chaque itération. Cependant, la LNS permet d'explorer dans le voisinage de solutions non améliorantes lorsque de telles solutions sont acceptées dans la procédure de recherche. Le but de cette stratégie est d'éviter la convergence prématurée vers des optima locaux et de forcer la procédure de recherche à visiter différentes zones de l'espace de solutions. Alors que l'heuristique LNS utilise seulement une procédure de destruction et une procédure de réparation, l'ALNS utilise plusieurs heuristiques de suppression et d'insertion et les applique alternativement en utilisant un mécanisme de sélection qui considère les statistiques obtenues pendant la recherche, d'où l'utilisation du terme *Adaptative*.

Pour la procédure de destruction, nous proposons 3 heuristiques de suppression :

1. **Heuristique de suppression de Shaw :** L'idée principale est de supprimer les clients "similaires", afin que nous puissions obtenir de nouvelles solutions après la réinsertion. En raison de la structure serrée des solutions du VRPTW, la suppression de clients très différents pourrait forcer la fonction de réparation à réinsérer ces clients à leurs positions d'origine.
2. **Heuristique de suppression aléatoire :** sélectionne simplement q clients de manière aléatoire et les supprime de la solution.
3. **Heuristique de suppression des clients mal-placés :** Dans cette heuristique, un coût marginal du service dans la solution courante est calculé pour chaque client. A chaque itération, le client avec le coût marginal le plus élevé est sélectionné.

Pour la procédure d'insertion, nous proposons 4 heuristiques d'insertion :

1. **Heuristique d'insertion simple :** est une heuristique de construction simple. Elle consiste à affecter chaque client à une nouvelle route.
2. **Heuristique d'insertion gloutonne :** consiste à insérer à chaque itération le client avec le coût d'insertion minimum.
3. **Heuristique d'insertion de *Regret* :** c'est une amélioration de l'heuristique gloutonne en intégrant une sorte d'anticipation pour le coût d'insertion de chaque client. Pour ce faire, une valeur de *regret* est calculée pour chaque client. Cette valeur mesure, à chaque itération, le regret de ne pas insérer le client dans sa meilleure route et de l'insérer dans sa deuxième meilleure route.
4. **Heuristique d'insertion non-myope :** L'idée principale de cette heuristique est d'évaluer le coût d'insertion d'un client en tenant compte de son impact sur les prochaines insertions.

Pour définir les deux heuristiques de suppression d'insertion à utiliser à chaque itération, nous utilisons un mécanisme de sélection qui permet d'utiliser les statistiques portant sur l'efficacité des différentes heuristiques durant les dernières itérations.

Afin d'évaluer l'efficacité de la méthode proposée, nous avons réalisé une étude expérimentale approfondie. Nous avons utilisé un ensemble large avec plusieurs types d'instances. Les solutions obtenues avec la méthode heuristique sont comparées à celles obtenues avec l'algorithme de branch-and-price présenté dans la section A.3.

Les résultats obtenus montrent l'efficacité de notre heuristique. Des solutions quasi-optimales sont obtenues pour toutes les instances testées dans des temps de calcul raisonnables. Des solutions optimales pour 116 parmi 175 instances contenant jusqu'à 75 clients. Le gap

entre la solution heuristique et la solution optimale est d'environs 2% dans le pire cas. En termes de temps de calcul, la méthode d'heuristique est en moyenne 42 fois plus rapide que l'algorithme de branch-and-price. Pour les instances avec 50 clients, le temps de calcul moyen est de 19 secondes avec la méthode heuristique alors qu'il est supérieur à 1100 secondes avec la méthode exacte.

A.5 Algorithme de Branch-and-price pour le VRPTW sur le graphe du réseau routier

Dans la littérature, une deuxième approche est proposée afin de conserver l'ensemble de l'espace de solutions en adressant un problème de tournées de véhicules avec plusieurs attributs. Cette approche consiste à résoudre le problème directement sur un graphe simulant le réseau routier. Dans ce graphe, les arcs représentent les segments de routes et les nœuds représentent les jonctions de routes.

Un article clé à cet égard est proposé par Letchford et al. [89]. Leur objectif est de montrer qu'un algorithme de branch-and-price appliqué sur un graphe de réseau routier serait plus efficace que quand il est appliqué sur un multigraphe. Letchford et al. [89] soulignent l'impact de la modélisation par graphe de réseau routier sur un algorithme de branch-and-price. Mais, seul l'algorithme de génération de colonnes a été examiné. Dans cet étude, ils considèrent le problème de multiple voyageurs de commerce avec fenêtres de temps (m-TSPTW) et ils proposent une étude expérimentale avec des instances réalistes pour appuyer leur résultats.

Bien que les résultats obtenus par Letchford et al. [89] soient intéressants, leurs conclusions peuvent difficilement être généralisées pour plusieurs raisons. Tout d'abord, ils se sont basés sur des résultats obtenus pour un problème particulier qui est le m-TSPTW. Dans ce problème, une fenêtre de temps est associée à chaque client et aucune restriction sur la charge totale transportée le long d'une route n'est considérée. Pourtant, dans la pratique, les véhicules utilisés ont une capacité limitée et une demande (à livrer ou à ramasser) doit être servie pour chaque client. Ce problème pourrait avoir un impact significatif sur l'efficacité des algorithmes avec les approches du réseau routier et du multigraphe. Deuxièmement, leurs expériences sont basées sur des instances avec des densités de clients relativement élevées: deux ensembles d'instances sont considérés avec des densités égales, respectivement, à 33% et 66%. Les applications de la vie réelle sont définies sur des réseaux routiers à grande échelle dans lesquels un petit nombre de nœuds sont associés aux emplacements des clients. Troisièmement, ils étudient uniquement le sous-problème de génération de colonnes lorsque des routes non élémentaires (c'est à dire qu'un client peut être servi plusieurs fois) sont autorisés. Le cas avec seulement des routes élémentaires n'est pas exploré. De plus, il n'est pas garanti que les valeurs des solutions issues de la génération de colonnes avec les deux

approches soient les mêmes lorsque les routes non élémentaires sont autorisées. Ce problème n'est pas examiné dans leurs expérimentations et ils se concentrent uniquement sur les temps de calcul obtenus avec les deux algorithmes. Enfin, Letchford et al. [89] n'explorent que le sous-problème de génération de colonnes et ne conçoit pas de règles de branchement appropriées. Il convient de mentionner que les règles de branchement standard pour les problèmes de routage des véhicules peuvent être facilement adaptées pour gérer les arcs parallèles dans la représentation par multigraphe (voir [8]), mais ce n'est pas le cas de l'algorithme qui fonctionne directement sur le graphe du réseau routier. Des règles de branchement appropriées pour chaque modélisation peuvent résulter en deux schémas de branch-and-price différents. Ainsi, les conclusions basées sur les résultats obtenus (par génération de colonnes) au nœud racine ne peuvent pas être généralisées pour le schéma de branch-and-price complet. Pour toutes ces raisons, d'autres analyses et des comparaisons approfondies sur l'efficacité des algorithmes avec les deux approches de modélisations sont nécessaires pour parvenir à des conclusions globales.

Dans ce chapitre, nous proposons d'étudier plus en profondeur l'efficacité relative des algorithmes de branch-and-price avec l'approche du réseau routier et l'approche du multigraphe et d'analyser davantage les résultats reportés par Letchford et al. [89]. Nous nous intéressons à ce qui est probablement le problème de tournées de véhicules le plus simple et le plus étudié avec deux attributs: le VRPTW. Nous développons un algorithme de branch-and-price complet basé sur les algorithmes de génération de colonnes présentés dans [89]. Nous basons nos expériences sur trois types d'instances: (1) les instances générées par Letchford et al. [89]; (2) un ensemble d'instances réalistes construites en utilisant la même procédure utilisé par Letchford et al. [89]; (3) des instances dérivées de réseaux routiers réels. Une étude expérimentale approfondie est proposée afin d'analyser l'impact de différents facteurs (contraintes de capacité, densité de clients, etc.) sur l'efficacité relative de l'algorithme de branch-and-price.

Pour la procédure de génération de colonnes, la seule différence entre l'algorithme de branch-and-price basé sur le graphe du réseau routier et l'algorithme standard est qu'un nœud client peut être visité plusieurs fois même quand la contrainte sur les routes élémentaires est considérée; dans ce cas, le client est servi une fois et pour les autres passages le nœud client est considéré comme une nœud intermédiaire. Pour résoudre le sous-problème de génération de colonnes, nous adaptons un algorithme de programmation dynamique proposé pour résoudre le problème de plus court chemin élémentaire avec des contraintes de ressources (ESPPRC) proposé dans [53]. Dans l'algorithme proposé, quand un label est étendu à un nœud client deux nouveaux labels sont générés: dans le premier, le client est servi, et dans le deuxième le nœud client est considéré comme une jonction de routes et le client n'est pas servi.

Pour le schéma de branch-and-price, un problème majeur avec les règles de branchement standard se pose en utilisant la modélisation par graphe du réseau routier. Ce problème

vient du fait qu'une solution de routage fractionnaire peut être supporté par un flux entier. Pour faire face à ce problème, nous avons proposé un schéma de branchement spécifique afin de garantir l'intégrité de la solution de routage. En effet, quand la matrice des flux est fractionnaire, on branche sur un arc avec un flux fractionnaire. Quand le flux est entier mais la solution de routage est fractionnaire, on génère deux nouvelles branches :

- Dans la première branche, on commence par générer toutes les routes faisables sur le sous-graphe induit par la matrice de flux. Ensuite, nous résolvons un problème de partitionnement en se basant sur l'ensemble des routes obtenues.
- Dans la deuxième branche, nous éliminons la solution fractionnaire en imposant l'utilisation d'un arc qui n'est pas dans le sous-graphe induit par la matrice de flux.

Afin de compléter les résultats présentés par Letchford et al. [89] et pour en tirer des conclusions exhaustives, nous proposons le plan d'expérimentations suivant. Dans tous les cas, on compare les résultats obtenus sur le graphe du réseau routier et sur le multigraphe:

1. Nous commençons par comparer les résultats obtenus en résolvant la relaxation LP en utilisant la génération de colonnes où des routes non élémentaires sont autorisées pour le m-TSPTW;
2. Ensuite, nous explorons le cas où seules les routes élémentaires sont autorisées. Il est à noter que dans le m-TSPTW, nous ne prenons pas en compte les demandes des clients et les contraintes de capacité des véhicules: la faisabilité d'une route est évaluée en ne tenant compte que des contraintes de fenêtres de temps;
3. Puis, nous étudions l'impact des contraintes de capacité dans le VRPTW sur les performances des algorithmes de génération de colonnes;
4. Enfin, nous explorons l'algorithme complet de branch-and-price.

Dans tous les cas, les résultats obtenus montrent que l'algorithme de branch-and-price est plus efficace quand il est appliqué sur le multigraphe que quand il est appliqué sur le graphe du réseau routier. Par exemple, pour l'algorithme de branch-and-price complet 228 (parmi 300 instances réalistes testées) instances sont résolues en utilisant l'approche du multigraphe alors qu'en utilisant l'approche du réseau routier seules 191 instances sont résolues au bout de $2h$.

A.6 Problème de tournées de véhicules dépendant du temps avec des fenêtres de temps et informations sur le réseau routier

Dans les applications réelles de problèmes de tournées de véhicules, les temps de parcours sont soumis à des variations importantes au cours de la journée. Ces variations peuvent être dues à des événements prévisibles tels que des embouteillages et des événements imprévisibles tels que des accidents, des conditions météorologiques et des pannes de véhicules. Dans la littérature, ces variations sont traitées en utilisant des variantes dépendant du temps des problèmes de tournées des véhicules. Dans de telles variantes, à chaque segment de route est associée une fonction de temps qui indique le temps de trajet pour chaque date de départ.

Typiquement, les problèmes de tournées de véhicules dépendant du temps (TDVRPs) sont traités en utilisant la modélisation par un graphe complet du réseau routier. Comme pour le cas non-dépendant du temps, cette approche peut avoir des effets négatifs sur la qualité de la solution dans plusieurs situations, par exemple, quand plusieurs attributs sont définis sur les segments de routes. Pour éviter ses effets négatifs, deux approches sont a priori envisageables pour le cas des TDVRPs: la première consiste à représenter le réseau routier avec un multigraphe et la deuxième approche consiste à résoudre le problème directement sur un graphe du réseau routier. Cependant, en raison de la dépendance temporelle, il est difficile, voire impossible, de représenter le réseau routier avec un multigraphe. En effet, il faudrait calculer l'ensemble des chemins efficaces pour chaque paire de points d'intérêt et pour chaque date de départ possible, ce qui induit la résolution de nombreux problèmes NP-difficiles et l'utilisation de structures de données complexes. Pour cette raison, nous proposons d'aborder le problème sur le graphe du réseau routier. Nous sélectionnons le problème de tournées de véhicules dépendant du temps avec des fenêtres de temps (TDVRPTW) comme problème pilote. Dans ce problème, un graphe du réseau routier est donné, avec des fonctions de vitesse de déplacement affectées à chaque arc. Nous développons un algorithme de branch-and-price capable de résoudre exactement le TDVRPTW. Nous comparons nos solutions avec celles trouvées sur deux graphes complets issus du graphe route-réseau: un graphe min-cost où les chemins sont sélectionnés en fonction de leur distance de parcours, un graphe min-time où les chemins sont sélectionnés en fonction de leur temps de parcours. Nous basons nos expériences sur deux types d'instances: (1) les instances de référence simulant de petits réseaux routiers réels; (2) les instances dérivées d'un grand réseau routier réel.

Il est important de souligner que les détails sur comment les informations sur le temps de trajet peuvent être obtenues pour la modélisation par graphe complet sont absents dans la plupart des articles sur les VRP dépendant du temps. Classiquement, dans ces articles, le graphe complet est introduit en premier. Ensuite, des profils de vitesse variables dans le temps sont introduits sur chaque arc. Ces profils de vitesse sont finalement convertis en temps de déplacement. Avec cette approche, les vitesses ou les temps de déplacement ne sont pas

explicitement définis sur les arcs du réseau routier. De plus, il suppose implicitement que la vitesse est constante sur tous les arcs du réseau routier qui composent un arc dans le graphe client. Dans la littérature, seuls quelques articles s'intéressent à la façon dont les fonctions de vitesse de déplacement et de temps peuvent être obtenues à partir des informations sur le réseau routier. Afin d'évaluer correctement les fonctions de temps de parcours pour nos deux graphe basés sur les clients, nous développons deux algorithmes spécifiques. Les deux algorithmes commencent avec les mêmes informations que celles du TDVRPTW, c'est-à-dire les fonctions de vitesse de déplacement associées aux arcs du graphe réseau-routier.

Dans le min-cost graphe, les arcs représentent les chemins les plus courts (en distance) reliant les points d'intérêt (dépôts et emplacements des clients). Tous ces chemins peuvent être calculés facilement en appliquant $|C|+1$ fois (avec C est l'ensemble des clients) un algorithme de Dijkstra dans le graphe du réseau routier, successivement à partir du dépôt et des nœuds clients. Cependant, pour chaque chemin, nous devons également calculer la fonction de date d'arrivée associée.

Dans le min-time graphe, les arcs représentent les chemins les plus rapides reliant les points d'intérêt. Ils sont décrits par deux fonctions: une fonction de date d'arrivée et une fonction de distance. Notez qu'une fonction de distance est nécessaire car le chemin le plus rapide peut changer en fonction de la date de départ. Les chemins peuvent être calculés en appliquant $|C|+1$ fois un algorithme d'étiquetage dans le graphe du réseau routier.

Pour résoudre le TDVRPTW dans le graphe du réseau routier, nous avons développé un algorithme de branch-and-price. L'algorithme proposé dans ce chapitre est basé sur le schéma de branch-and-price présenté dans le chapitre précédant pour le même problème avec des temps de parcours constants. Par rapport à cet algorithme, deux modifications principales ont été effectuées. Tout d'abord, les temps de parcours sont calculés en fonction des dates de départ lors de l'extension des étiquettes dans le sous-problème de génération de colonnes. Deuxièmement, un algorithme de la programmation dynamique bidirectionnelle a été implémentée pour résoudre le sous-problème de génération de colonnes.

Pour évaluer l'impact de notre approche sur la qualité de la solution, nous avons effectué une étude expérimentale en nous basant sur un ensemble large d'instances (instances réalistes simulant des petits réseaux routiers et instances construites à partir des données d'un réseau routier réel avec un grand nombre de nœuds).

Les résultats obtenus montrent qu'en adressant le problème directement sur le graphe du réseau routier au lieu d'utiliser la modélisation par graphe complet, le coût de la solution est réduit jusqu'à 12.4% comparée à la solution obtenue dans le min-cost graphe et jusqu'à 20% comparée à la solution obtenue dans le min-time graphe. Les gains moyens en coût de la solution sont de 1.7% et 7.3%. En termes de temps de calcul, nous remarquons que l'algorithme de branch-and-price est moins efficace quand il est appliqué au problème dans

le graphe du réseau routier que quand il est appliqué au problème basé sur la modélisation par graphe complet. Nous remarquons aussi que quand le nombre de clients augmente le temps de calcul de l'algorithme de branch-and-price sur le réseau routier augmentent significativement. Pour les instances réelles, le temps de calcul moyen est de 13.2 secondes pour les instances avec 5 clients et atteint 63.5 secondes quand le nombre de clients est égal à 25.

A.7 Conclusions et perspectives

Les problèmes de tournées de véhicules constituent l'une des classes les plus étudiées des problèmes d'optimisation combinatoire. Cela est dû au grand nombre d'applications réelles impliquant des problèmes de tournées.

Classiquement, ces problèmes sont abordés en utilisant la modélisation par un graphe dit basé sur les clients, un graphe complet représentant le réseau routier. Dans de nombreuses situations, cette modélisation peut avoir d'importantes conséquences. Une situation est lorsque plusieurs attributs sont définis sur des segments de route. Dans ce cas, des chemins alternatifs avec des compromis différents ne sont pas pris en compte dans le graphe basé sur les clients. Cela peut avoir un impact négatif sur la qualité de la solution.

La première contribution de cette thèse est d'analyser des travaux où les limites de la représentation par un graphe complet sont évoquées et qui étudient les problèmes de tournées de véhicules avec plus d'informations sur le réseau routier. Nous examinons les différents cas où le graphe complet représente mal le réseau routier et nous mettons en évidence les approches alternatives proposées pour gérer ces limites. L'étude montre le manque de contributions dans ce domaine et révèle de nombreuses directions de recherche inexplorées.

Dans la littérature, deux approches alternatives sont proposées pour gérer les limites de la modélisation par graphe basé sur les clients lorsque plusieurs attributs sont définis sur des segments de route. Dans la première approche, le réseau routier est représenté en utilisant un multigraphe où un arc est introduit pour chaque chemin alternatif. Dans la deuxième approche, le problème est résolu directement sur un graphe représentant le réseau routier.

Une première partie de cette thèse porte sur l'approche multigraphe. Nous étudions la possibilité de représenter des réseaux routiers réels avec des multigraphes. Nous développons une méthode de résolution exacte qui calcule l'ensemble des chemins non dominés reliant un ensemble de points d'intérêt dans un réseau routier. L'algorithme proposé est basé sur une stratégie de recherche A* multi-destinations multi-objectif. En utilisant la méthode proposée, des multigraphes avec jusqu'à 500 points d'intérêt sont construits en quelques secondes pour les grands réseaux routiers. Une analyse sur l'impact de la densité de clients dans le réseau routier est réalisée. En outre, nous étudions l'impact des fenêtres de temps des

clients sur le nombre de chemins alternatifs réalisables et sur le temps de construction.

Dans la deuxième étape, nous étudions l'impact de la représentation par multigraphe sur la qualité de la solution pour un problème de tournées de véhicules avec deux attributs: le VRPTW. Nous développons un algorithme de branch-and-price. Une étude expérimentale approfondie est réalisée sur des instances modifiées de la littérature et sur des instances dérivés de réseaux routiers réels. Les résultats montrent que, en utilisant le multigraphe, les coûts de la solution sont significativement réduits par rapport aux coûts de la solution sur le graphe complet avec les plus court chemins (jusqu'à 14%) et comparés aux coûts de la solution sur le graphe complet avec les chemins les plus rapides (jusqu'à 54%). De plus, nous remarquons que l'utilisation du multigraphe n'augmente que légèrement les temps de calcul par rapport au graphe basé sur les clients. Dans une autre contribution, nous développons une méthode heuristique qui permet de résoudre efficacement la variante multigraphe du problème. La méthode proposée est basée sur une recherche adaptative de voisinages larges capable d'explorer différentes zones de l'espace des solutions. Nous intégrons dans l'algorithme une structure de données incrémentale et une procédure basée sur la programmation dynamique qui permettent d'évaluer efficacement le voisinage d'une solution donnée en présence d'arcs parallèles entre les nœuds. Les expérimentations montrent la compétitivité de l'algorithme développé comparé à l'algorithme de branch-and-price.

Une deuxième partie de la thèse porte sur l'approche basée sur le réseau routier. Nous étudions d'abord l'efficacité relative de la résolution du problème directement sur le réseau routier par rapport à l'approche par multigraphe. Pour ce faire, nous développons un algorithme de branch-and-price qui fonctionne directement sur le réseau routier. Une étude expérimentale approfondie est ensuite réalisée afin d'analyser l'impact des caractéristiques du VRPTW (routes élémentaires et non élémentaires, fenêtres de temps des clients, contraintes de capacité, densité de clients, etc.) sur la performance des algorithmes de branch-and-price avec le graphe du réseau routier et dans le multigraphe. Les résultats obtenus montrent qu'un schéma de branch-and-price basé sur le multigraphe est plus efficace que l'approche du réseau routier.

La deuxième contribution de cette partie de la thèse est de souligner les limites du graphe basé sur les clients pour le VRPTW dépendant du temps. Pour ce faire, nous développons d'abord deux algorithmes pour calculer deux représentations par graphe complet du réseau routier. Ensuite, nous adaptons l'algorithme de branch-and-price basé sur le réseau routier pour gérer le paramètre dépendant du temps. Les résultats montrent les avantages potentiels de la modélisation par un graphe du réseau routier et les économies attrayantes en termes de coût de la solution obtenue. Il apparaît également que des solutions optimales sur le réseau routier sont trouvées pour des instances qui ne sont pas réalisables en utilisant la représentation par graphe complet basée sur les chemins les plus courts en distance.

L'intérêt principal de cette thèse est de souligner les limites de la modélisation standard

utilisée pour s'attaquer à une grande classe de problèmes de tournées de véhicules. Nous nous concentrons sur les problèmes où plusieurs attributs sont définis sur des segments de route. Cependant, il existe de nombreuses autres situations où la modélisation avec le graphe complet n'est pas adaptée pour résoudre efficacement les problèmes de tournées des véhicules. Au moment de la conclusion de cette thèse, seuls quelques articles étudient l'impact de la résolution de problèmes à l'aide de la modélisation classique. Une extension naturelle du travail présenté dans cette thèse serait d'examiner ces situations et d'analyser les avantages de procéder différemment que d'utiliser l'approche de modélisation standard.

D'un point de vue méthodologique, des travaux de recherche futurs pourraient s'intéresser au développement d'approches de résolution efficaces capables de gérer le graphe du réseau routier. Nous rappelons que notre objectif dans les chapitres 6 et 7 n'était pas d'atteindre la meilleure implémentation possible de l'algorithme de branch-and-price pour le VRPTW basé sur le réseau routier, mais nous avons cherché à développer une méthode qui permette d'obtenir des résultats concluants. L'algorithme proposé peut être amélioré de plusieurs façons. Premièrement, on aimerait améliorer la règle de branchement. Dans le schéma de branchement proposé au chapitre 6, les décisions ont principalement un impact sur le problème maître et la qualité des bornes inférieures, mais n'affectent pas le sous-problème de génération de colonnes. De plus, ces décisions seules ne garantissent pas l'intégrité de la solution. Il serait intéressant de concevoir des règles de branchement permettant d'obtenir une solution entière et ayant un impact sur la structure du problème de génération de colonnes. Une deuxième amélioration possible est la conception de règles de dominance spécifiques pour les paramètres du graphe du réseau routier. Dans notre implémentation, nous utilisons la règle de dominance introduite pour le VRPTW standard. L'utilisation de règles de branchement plus appropriées améliorerait significativement l'efficacité de l'algorithme de génération de colonnes. De plus, il serait intéressant de mettre en œuvre un test d'élimination qui élimine les étiquettes non dominées qui, en s'étendant de la meilleure façon, n'améliorent pas la solution actuelle.

Une autre direction de recherche intéressante serait de concevoir une méthode heuristique capable de gérer les paramètres du graphe du réseau routier. La plupart des approches heuristiques proposées pour les problèmes de tournées de véhicules sont basées sur des mouvements de recherche local qui visent à déplacer les services des clients pour explorer de nouvelles solutions. En utilisant la modélisation par un graphe du réseau routier, de telles opérations ne sont pas faciles à évaluer et à effectuer car les nœuds clients ne sont pas directement connectés et plusieurs chemins peuvent exister entre deux points d'intérêt. De nombreux autres défis de calcul se posent dans ce cas.

De plus, les progrès récents des technologies de l'information et de la communication permettent d'obtenir des informations sur les conditions de circulation et les temps de déplacement en temps réel. Bien que l'on se soit de plus en plus intéressé à ces informations lorsqu'on s'attaque aux problèmes de planification des trajets et des itinéraires les plus courts,

les problèmes de tournées manquent dans ce domaine. Il serait intéressant d'examiner l'effet de considérer de telles informations pour les problèmes de tournées de véhicules et d'étudier l'impact des graphes basés sur les clients et des approches de graphe du réseau routier sur la qualité de la solution pour ces problèmes. Il serait également intéressant d'examiner s'il est suffisant de travailler avec la représentation par graphe basé sur des clients pour d'autres catégories de problèmes de tournées de véhicules tels que les problèmes avec possibilité de déviation, les problèmes avec des contraintes de synchronisation, le routage des matériaux dangereux, etc. Dans ces problèmes, plusieurs questions doivent être considérées, par exemple la probabilité d'incident et l'exposition de la population pour le transport de matières dangereuses, l'attribution de nouvelles demandes de service en fonction des situations réelles des véhicules, etc. Ainsi, il serait intéressant de proposer des approches de modélisation alternatives pour de tels problèmes.

List of Tables

List of Figures

Bibliography

[1] Juancarlo Anez, Tomás De La Barra, and Beatnz Pérez. Dual graph representation of transport networks. *Transportation Research Part B: Methodological*, 30(3):209–216, 1996.

[2] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

[3] Roberto Baldacci, Lawrence Bodin, and Aristide Mingozzi. The multiple disposal facilities and multiple inventory locations rollon–rolloff vehicle routing problem. *Computers & Operations Research*, 33(9):2667–2702, 2006.

[4] Jonathan F Bard, George Kontoravdis, and Gang Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36(2):250–269, 2002.

[5] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[6] JE Beasley. Adapting the savings algorithm for varying inter-customer travel times. *Omega*, 9(6):658–659, 1981.

[7] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, and Alain Quilliot. Adaptive large neighborhood search for the vehicle routing problem with time windows with a multigraph representation for the road network. Technical Report EMSE CMP-–SFL 2017/7, Ecole des Mines de Saint Etienne, CMP, Gardanne France, 2017.

[8] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, and Alain Quilliot. Empirical analysis for the vrptw with a multigraph representation for the road network. *Computers & Operations Research*, 2017.

[9] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, and Alain Quilliot. A solution method for the multi-destination bi-objectives shortest path problem. Technical Report EMSE CMP-–SFL 2017/5, Ecole des Mines de Saint Etienne, CMP, Gardanne, France, 2017.

[10] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, and Alain Quilliot. Vehicle routing problems with road-network information: State of the art. *Networks*, 2018.

[11] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, Alain Quilliot, and Tom Van Woensel. A branch-and-price algorithm for the vehicle routing problem with time windows on a road-network graph. Technical Report EMSE CMP-–SFL 2017/9, Ecole des Mines de Saint Etienne, CMP, Gardanne, France, 2017.

[12] Claudia Bode and Stefan Irnich. Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations research*, 60(5):1167–1182, 2012.

[13] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

[14] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118, 2005.

[15] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139, 2005.

[16] J Brumbaugh-Smith and D Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2):216–224, 1989.

[17] M Caramia and F Guerriero. A heuristic approach to long-haul freight transportation with multiple objective functions. *Omega*, 37(3):600–614, 2009.

[18] Alain Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006.

[19] Maxime Chassaing, Christophe Duhamel, and Philippe Lacomme. Time-dependent vehicle routing problem with waiting times. In *Odysseus 2015*, pages 197–200, 2015.

[20] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[21] João Carlos Namorado Climaco and Ernesto Queiros Vieira Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4):399–404, 1982.

[22] João CN Clímaco and Marta Pascoal. Multicriteria path and tree problems: discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19(1-2):63–98, 2012.

[23] Ángel Corberán and Gilbert Laporte. *Arc routing: problems, methods, and applications*, volume 20. SIAM, 2015.

[24] Angel Corberán and Christian Prins. Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69, 2010.

[25] J. F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. Vrp with time windows. In *The vehicle routing problem*, pages 157–193. Society for Industrial and Applied Mathematics, 2001.

[26] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

[27] Jean-François Cordeau, Gianpaolo Ghiani, and Emanuela Guerriero. Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science*, 48(1):46–58, 2012.

[28] H William Corley and I Douglas Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Applications*, 46(1):79–86, 1985.

[29] Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical programming*, 33(1):1–27, 1985.

[30] Said Dabia, Stefan Ropke, Tom Van Woensel, and Ton De Kok. Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation science*, 47(3):380–396, 2013.

[31] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[32] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[33] Bruno De Backer, Vincent Furnon, P Prosser, P Kilby, and Paul Shaw. Local search in constraint programming: Application to the vehicle routing problem. In *Proc. CP-97 Workshop Indust. Constraint-Directed Scheduling*, pages 1–15. Schloss Hagenberg Austria, 1997.

[34] Leizer de Lima Pinto, Cláudio Thomás Bornstein, and Nelson Maculan. The tricriterion shortest path problem with at least two bottleneck objective functions. *European Journal of Operational Research*, 198(2):387–391, 2009.

[35] Daniel Delling and Dorothea Wagner. Time-dependent route planning. *Robust and online large-scale optimization*, 5868(1):207–230, 2009.

[36] Sofie Demeyer, Jan Goedgebeur, Pieter Audenaert, Mario Pickavet, and Piet Demeester. Speeding up martins' algorithm for multiple objective shortest path problems. *4OR*, 11(4):323–348, 2013.

[37] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[38] Guy Desaulniers, François Lessard, and Ahmed Hadjar. *Tabu search, generalized k-path inequalities, and partial elementarity for the vehicle routing problem with time windows*. Groupe d'études et de recherche en analyse des décisions, 2006.

[39] Guy Desaulniers, Oli BG Madsen, and Stefan Ropke. The vehicle routing problem with time windows. *Vehicle routing: Problems, methods, and applications*, 18:119–159, 2014.

[40] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[41] Jacques Desrosiers, Yvan Dumas, Marius M Solomon, and François Soumis. Time constrained routing and scheduling. *Handbooks in operations research and management science*, 8:35–139, 1995.

[42] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[43] Alberto V Donati, Roberto Montemanni, Norman Casagrande, Andrea E Rizzoli, and Luca M Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European journal of operational research*, 185(3):1174–1191, 2008.

[44] Moshe Dror. *Arc routing: theory, solutions and applications*. Springer Science & Business Media, 2012.

[45] Olivier Du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[46] EEA. Evaluating 15 years of transport and environmental policy integration. term 2015: Transport indicators tracking progress towards environmental targets in europe, eea report 7/2015. Technical report, European Environment Agency, 2015.

[47] Richard Eglese, Will Maden, and Alan Slater. A road timetabletm to aid vehicle routing and scheduling. *Computers & operations research*, 33(12):3508–3519, 2006.

[48] Jan Fabian Ehmke, Ann Melissa Campbell, and Barrett W Thomas. Data-driven approaches for emissions-minimized paths in urban areas. *Computers & Operations Research*, 67:34–47, 2016.

[49] Jan Fabian Ehmke, Ann Melissa Campbell, and Barrett W Thomas. Vehicle routing to minimize time-dependent emissions in urban areas. *European Journal of Operational Research*, 251(2):478–494, 2016.

[50] Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

[51] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[52] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[53] Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[54] Miguel Andres Figliozzi. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review*, 48(3):616–636, 2012.

[55] Marshall Fisher. Vehicle routing. *Handbooks in operations research and management science*, 8:1–33, 1995.

[56] Bernhard Fleischmann. A cutting plane procedure for the travelling salesman problem on road networks. *European Journal of Operational Research*, 21(3):307–317, 1985.

[57] Bernhard Fleischmann, Martin Gietz, and Stefan Gnutzmann. Time-varying travel times in vehicle routing. *Transportation science*, 38(2):160–173, 2004.

[58] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[59] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

[60] Xavier Gandibleux, Frédéric Beugnies, and Sabine Randriamasy. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR*, 4(1):47–59, 2006.

[61] Thierry Garaix, Christian Artigues, Dominique Feillet, and Didier Josselin. Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research*, 204(1):62–75, 2010.

[62] Michel Gendreau, Gianpaolo Ghiani, and Emanuela Guerriero. Time-dependent routing problems: A review. *Computers & operations research*, 64:189–197, 2015.

[63] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994.

[64] Keivan Ghoseiri and Behnam Nadjari. An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing*, 10(4):1237–1246, 2010.

[65] B. L. Golden, S. Raghavan, and E. A. Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[66] F Guerriero and R Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of optimization theory and applications*, 111(3):589–613, 2001.

[67] Pierre Hansen. Bicriterion path problems. In *Multiple criteria decision making theory and application*, pages 109–127. Springer, 1980.

[68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[69] Markus Hesse and Jean-Paul Rodrigue. The transport geography of logistics and freight distribution. *Journal of transport geography*, 12(3):171–184, 2004.

[70] Bo Huang, Li Yao, and K Raguraman. Bi-level ga and gis for multi-objective tsp route planning. *Transportation planning and technology*, 29(2):105–124, 2006.

[71] Yixiao Huang, Lei Zhao, Tom Van Woensel, and Jean-Philippe Gross. Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological*, 95:169–195, 2017.

[72] F Huarng, PS Pulat, and L Shih. A computational comparison of some bicriterion shortest path algorithms. *Journal of the Chinese Institute of Industrial Engineers*, 13(2):121–125, 1996.

[73] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Vehicle dispatching with time-dependent travel times. *European journal of operational research*, 144(2):379–396, 2003.

[74] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. *Column Generation*, 6730:33–65, 2005.

[75] Stefan Irnich and Daniel Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for k $\geq$ 3. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

[76] O Jabali, T Woensel, and AG de Kok. Analysis of travel times and co2 emissions in time-dependent vehicle routing. *Production and Operations Management*, 21(6):1060–1074, 2012.

[77] Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. *Oper. Res. Forthcoming*, 2006.

[78] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330, 2008.

[79] Brian Kallehauge, Jesper Larsen, and Oli BG Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33(5):1464–1487, 2006.

[80] David E Kaufman and Robert L Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1(1):1–11, 1993.

[81] Astrid S Kenyon and David P Morton. Stochastic vehicle routing with random travel times. *Transportation Science*, 37(1):69–82, 2003.

[82] Oanh Tran Thi Kim, VanDung Nguyen, Seung Il Moon, and Choong Seon Hong. Finding realistic shortest path in road networks with lane changing and turn restriction. In *Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific*, pages 1–4. IEEE, 2016.

[83] Niklas Kohl, Jacques Desrosiers, Oli BG Madsen, Marius M Solomon, and Francois Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.

[84] Adrianus Leendert Kok, Elias W Hans, and Johannes MJ Schutten. Vehicle routing under time-dependent travel times: the impact of congestion avoidance. *Computers & operations research*, 39(5):910–918, 2012.

[85] David SW Lai, Ozgun Caliskan Demirag, and Janny MY Leung. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transportation Research Part E: Logistics and Transportation Review*, 86:32–52, 2016.

[86] Zhifeng Lang, Enjian Yao, Weisong Hu, and Zheng Pan. A vehicle routing problem solution considering alternative stop points. *Procedia-Social and Behavioral Sciences*, 138:584–591, 2014.

[87] Gilbert Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

[88] Gilbert Laporte, Francois Louveaux, and Hélène Mercure. The vehicle routing problem with stochastic travel times. *Transportation science*, 26(3):161–170, 1992.

[89] Adam N Letchford, Saeideh D Nasiri, and Amar Oukil. Pricing routines for vehicle routing with time windows on road networks. *Computers & Operations Research*, 51:331–337, 2014.

[90] Adam N Letchford, Saeideh D Nasiri, and Dirk Oliver Theis. Compact formulations of the steiner traveling salesman problem and related problems. *European Journal of Operational Research*, 228(1):83–92, 2013.

[91] Adam N Letchford and Amar Oukil. Exploiting sparsity in pricing routines for the capacitated arc routing problem. *Computers & Operations Research*, 36(7):2320–2327, 2009.

[92] Qingquan Li, Bi Yu Chen, Yafei Wang, and William HK Lam. A hybrid link-node approach for finding shortest paths in road networks with turn restrictions. *Transactions in GIS*, 19(6):915–929, 2015.

[93] Xiangyong Li, Peng Tian, and Stephen CH Leung. Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, 125(1):137–145, 2010.

[94] Jens Lysgaard. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 175(1):210–223, 2006.

[95] Ernesto Queiros Vieira Martins. An algorithm for ranking paths that may contain cycles. *European Journal of Operational Research*, 18(1):123–130, 1984.

[96] Ernesto Queiros Vieira Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

[97] Ernesto Queiros Vieira Martins, Marta Margarida Braz Pascoal, and Jose Luis Esteves Dos Santos. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10(03):247–261, 1999.

[98] Ernesto Queros Vieira Martins and JLE Santos. The labeling algorithm for the multiobjective shortest path problem. *Departamento de Matematica, Universidade de Coimbra, Portugal, Tech. Rep. TR-99/005*, 1999.

[99] John Mote, Ishwar Murthy, and David L Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53(1):81–92, 1991.

[100] Matthias Müller-Hannemann and Karsten Weihe. On the cardinality of the pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147(1):269–286, 2006.

[101] Commission of the European Communities. sustainable future for transport: Towards an integrated, technology-led and user friendly system. Technical report, COM, 2009.

[102] Ariel Orda and Raphael Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM)*, 37(3):607–625, 1990.

[103] CS Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

[104] José Manuel Paixão and José Luis Santos. Labeling methods for the general case of the multi-objective shortest path problem–a computational study. In *Computational Intelligence and Decision Making*, pages 489–502. Springer, 2013.

[105] José Maria A Pangilinan and Gerrit K Janssens. Evolutionary algorithms for the multiobjective shortest path problem. *World Academy of Science, Engineering and Technology, International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, 1(1):7–12, 2007.

[106] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. 50th anniversary invited article—goods distribution with electric vehicles: review and research perspectives. *Transportation Science*, 50(1):3–22, 2016.

[107] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007.

[108] Christian Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

[109] Jiani Qian and Richard Eglese. Finding least fuel emission paths in a network with time-varying speeds. *Networks*, 63(1):96–106, 2014.

[110] Jiani Qian and Richard Eglese. Fuel emissions optimization in vehicle routing problems with time-varying speeds. *European Journal of Operational Research*, 248(3):840–848, 2016.

[111] Andrea Raith. Speed-up of labelling algorithms for biobjective shortest path problems. In *Proceedings of the 45th annual conference of the ORSNZ. Auckland, New Zealand*, pages 313–322, 2010.

[112] Andrea Raith and Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4):1299–1331, 2009.

[113] Line Blander Reinhardt, Mads Kehlet Jepsen, and David Pisinger. The edge set cost of the vehicle routing problem with time windows. *Transportation Science*, 50(2):694–707, 2015.

[114] Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006.

[115] Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

[116] Jean-Paul Rodrigue, Claude Comtois, and Brian Slack. *The geography of transport systems*. Routledge, 2009.

[117] Stefan Ropke and Jean-François Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009.

[118] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

[119] L. M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.

[120] D Schultes. Tiger road networks for 9th dimacs implementation challenge–shortest path, 2005.

[121] Paolo Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In *Recent advances and historical development of vector optimization*, pages 222–232. Springer, 1987.

[122] M Setak, Z Shakeri, and A Patoghi. A time dependent pollution routing problem in multi-graph. *International Journal of Engineering-Transactions B: Applications*, 30(2):234, 2017.

[123] Mostafa Setak, Majid Habibi, Hossein Karimi, and Mostafa Abedzadeh. A time-dependent vehicle routing problem in multigraph with fifo property. *Journal of Manufacturing Systems*, 35(35):37–45, 2015.

[124] Paul Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 1997.

[125] Anders JV Skriver. A classification of bicriterion shortest path (bsp) algorithms. *Asia Pacific Journal of Operational Research*, 17(2):199–212, 2000.

[126] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[127] Bradley S Stewart and Chelsea C White III. Multiobjective a. *Journal of the ACM (JACM)*, 38(4):775–814, 1991.

[128] P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. Siam, 2014.

[129] Paolo Toth and Daniele Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1):487–512, 2002.

[130] George Tsaggouris and Christos Zaroliagis. Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. *Theory of Computing Systems*, 45(1):162–186, 2009.

[131] Chi Tung Tung and Kim Lin Chew. A multicriteria pareto-optimal path algorithm. *European Journal of Operational Research*, 62(2):203–209, 1992.

[132] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013.

[133] HF Wang and YY Lee. Two-stage particle swarm optimization algorithm for the time dependent alternative vehicle routing problem. *Journal of Applied & Computational Mathematics*, 3(4):1–9, 2014.

[134] Liang Wen, Bülent Çatay, and Richard Eglese. Finding a minimum cost path between a pair of nodes in a time-varying road network with a congestion charge. *European Journal of Operational Research*, 236(3):915–923, 2014.

[135] Liang Wen and Richard Eglese. Minimum cost vrp with time-dependent speed data and congestion charge. *Computers & Operations Research*, 56:41–50, 2015.

[136] Stephan Winter. Modeling costs of turns in route planning. *GeoInformatica*, 6(4):345–361, 2002.

[137] Konstantinos G Zografos and Konstantinos N Androutsopoulos. A decision support system for integrated hazardous materials routing and emergency response decisions. *Transportation Research Part C: Emerging Technologies*, 16(6):684–703, 2008.

École Nationale Supérieure des Mines de Saint-Étienne

NNT: 820

Hamza BEN TICHA

Vehicle Routing Problems with road-network information

Keywords: Vehicle routing problems, road network, multigraph

Abstract:

Vehicle routing problems (VRPs) have drawn many researchers' attention for more than fifty years. Most approaches found in the literature are, implicitly, based on the key assumption that the best path between each two points of interest in the road network (customers, depot, etc.) can be easily defined. Thus, the problem is tackled using the so-called customer-based graph, a complete graph representation of the road network. In many situations, such a graph may fail to accurately represent the original road network and more information are needed to address correctly the routing problem.

We first examine these situations and point out the limits of the traditional customer-based graph. We propose a survey on works investigating vehicle routing problems by considering more information from the road network. We outline the proposed alternative approaches, namely the multigraph representation and the road network approach.

Then, we are interested in the multigraph approach. We propose an algorithm that efficiently compute the multigraph representation for large sized road networks. We present an empirical analysis on the impact of the multigraph representation on the solution quality for the VPR with time windows (VRPTW) when several attributes are defined on road segments. Then, we develop an efficient heuristic method for the multigraph-based VRPTW.

Next, we investigate the road network approach. We develop a complete branch-and-price algorithm that can solve the VRPTW directly on the original road network. We evaluate the relative efficiency of the two approaches through an extensive computational study.

Finally, we are interested in problems where travel times vary over the time of the day, called time dependent vehicle routing problems (TDVRPs). We develop a branch-and-price algorithm that solves the TDVRP with time windows directly on the road network and we analyse the impact of the proposed approach on the solution quality.

École Nationale Supérieure des Mines de Saint-Étienne

NNT : 820

Hamza BEN TICHA

Problèmes de tournées de véhicules avec des informations du réseau routier

Mots clefs : Problèmes de tournées de véhicules, réseau routier, multigraphe

Résumé :

Les problèmes de tournées de véhicules (VRPs) ont fait l'objet de plusieurs travaux de recherche depuis maintenant plus de 50 ans. La plupart des approches trouvées dans la littérature s'appuient sur un graphe complet ou un noeud est introduit pour tout point d'intérêt du réseau routier (typiquement les clients et le dépôt). Cette modélisation est, implicitement, basée sur l'hypothèse que le meilleur chemin entre toute paire de points du réseau routier est bien défini. Cependant, cette hypothèse n'est pas toujours valide dans de nombreuses situations. Souvent, plus d'informations sont nécessaires pour modéliser et résoudre correctement le problème.

Nous commençons par examiner ces situations et définir les limites de la modélisation basée sur un graphe complet. Nous proposons un état de l'art des travaux qui examinent ces limites et qui traitent des VRPs en considérant plus d'informations issues du réseau routier. Nous décrivons les approches alternatives proposées, à savoir la modélisation utilisant un multi-graphe et celle utilisant la résolution directe sur un graph représentant le réseau routier.

Dans une seconde étude, nous nous intéressons à l'approche basée sur la construction d'un multi-graphe. Nous proposons, d'abord, un algorithme qui permet de calculer d'une manière efficace la représentation par multi-graph du réseau routier. Puis, nous présentons une analyse empirique sur l'impact de cette modélisation sur la qualité de la solution. Pour ce faire, nous considérons le problème classique VRPTW comme un problème de pilote. Par la suite, nous développons une méthode heuristique efficace afin de résoudre le VRPTW basée sur une représentation par un multi-graphe.

Dans une troisième étape, nous nous concentrons sur l'approche basée sur la résolution directe du problème sur un graphe représentant le réseau routier. Nous développons un algorithme de type branch-and-price pour la résolution de cette variante du problème. Une étude expérimentale est, ensuite, menée afin d'évaluer l'efficacité relative des deux approches.

Enfin, nous étudions les problèmes de tournées de véhicules dans lesquels les temps de parcours varient au cours de la journée. Nous proposons un algorithme de type branch-and-price afin de résoudre le problème avec des fenêtres de temps directement sur le graphe représentant le réseau routier. Une analyse empirique sur l'impact de l'approche proposée sur la qualité de la solution est proposée.