Modèles et Algorithmes pour les Problèmes de Livraison du Dernier Kilomètre avec Plusieurs Options d'Expédition

Yuan Yuan

N° d'ordre: | 3 | 8 | 3 |

CENTRALE LILLE

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

en

Spécialité : Informatique

par

Yuan YUAN

DOCTORAT DELIVRE PAR CENTRALE LILLE

Titre de la thèse :

Modèles et Algorithmes pour les Problèmes de Livraison du Dernier Kilomètre avec Plusieurs Options d'Expédition

Models and Algorithms for Last Mile Delivery Problems with Multiple Shipping Options

---

Soutenue le 14 octobre 2019 devant le jury d'examen :

Président	M. Dominique Feillet	Professeur, Ecole des Mines de Saint-Etienne
Rapporteur	M. Yves Crama	Professeur, HEC Liège
Rapporteur	M. Guido Perboli	Professeur associé, Politecnico di Torino
Examinateur	Mme Claudia Archetti	Professeur associée, ESSEC Paris
Examinateur	Mme Martine Labbé	Professeur, Université Libre de Bruxelles
Examinateur	Mme Caroline Prodhon	Maître de conférences HDR, Université de Technologie de Troyes
Directeur de thèse	M. Frédéric Semet	Professeur, Centrale Lille
Invité	M. Diego Cattaruzza	Maître de conférences, Centrale Lille
Invité	M. Maxime Ogier	Maître de conférences, Centrale Lille

Thèse préparée dans le Laboratoire CRIStAL au sein de l'équipe INOCS à l'Inria Lille Nord-Europe
Ecole doctorale SPI 072 (Lille, Artois, ULCO, UPHF, Centrale Lille)

À mes parents,

à toute ma famille,

à mes professeurs,

et à mes chèr(e)s ami(e)s.

Acknowledgements

This thesis is the result of three years of work and it would not have been possible without the kind support and help from many brilliant individuals. I would like to express my sincere thanks to all of them.

First and foremost, I would like to give my wholehearted gratitude to my supervisors M. Frédéric Semet, M. Diego Cattaruzza and M. Maxime Ogier for their continuous support for my PhD study and research. Their immense knowledge, enthusiasm and guidance helped me through each stage of the research and writing of this thesis. I could not have imagined having better supervisors. Especially, I would like to express my deepest appreciation to Diego and Maxime for being there for me whenever I need. Thank you for every small meeting and discussion. Thank you for all the advises and help during the writing of articles. Thank you for guiding me with persistent patience. I am also grateful for the opportunities you have offered me to meet people and share my research. Not only you show me how to do research, but also the right attitude to be a researcher. You are such wonderful people and I feel really lucky to have you.

Besides my supervisors, I would like to thank the rest of my PhD jury committee: Prof. Yves Crama, Prof. Guido Perboli, Prof. Claudia Archetti, Prof. Dominique Feillet, Prof. Martine Labbé, and Prof. Caroline Prodhon. I appreciate the time you took to read my thesis, the insightful comments that helped improving it, and the interesting discussions we had during my defense.

I would also like to express my special gratitude to Prof. Daniele Vigo for accepting me to do a two-week academic visit in his team in University of Bologna, Italy. I am thankful that you shared with me your expertise and very useful ideas during the collaboration. It was great to have the opportunity to work with you.

I thank my fellow labmates in INOCS team: Wenjuan, Yaheng, Léo, Arnaud, Luis, Mathieu, Matteo, Youcef, Lilian, María, Conchi, Pablito, Pablo, Juan, Burak, Daniel,

ACKNOWLEDGEMENTS

Sebastien, Anicet, Tifaout for all the time that we spent together and all the fun we had in the last three years. I really enjoyed and appreciated all your accompany.

I would like to thank my boyfriend Anthony for his accompany, support, encouragements, patience and love. I am very lucky and happy to have you by my side. And it is nice that we have each other for the adventure of becoming a Doctor!

Last but not least, I would like to thank my parents and my whole family for supporting me throughout my life.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Introduction

Nowadays, e-commerce is a thriving market around the world. It is used on a daily basis and allows customers to purchase online whenever and whatever they like. Customers are no longer restricted to go to a specific store and to respect the opening hours. An annual survey conducted by the analytics firm comScore and UPS revealed that American consumers purchased more things online than in stores in 2016 (Farber, 2016). At the end of 2018, global e-commerce sales reached approximately $2.8 trillion and are estimated to hit $4.5 trillion in 2021 (Wardini, 2018). This growing e-commerce poses a huge challenge for the last mile delivery since the ordered items need to be delivered to individual customers.

Currently, there exist several last mile delivery services to deliver packages to customers. The most common delivery option is home/workplace delivery (Lowe & Rigby, 2014). Customers wait at home/workplace to get their packages. Besides, the delivery can be made to pick-up points such as dedicated lockers or stores. In this case, customers can retrieve their packages once the delivery has been made. To give an idea, there are more than 2800 lockers located across the US (Holsenbeck, 2018). When customers shop online, they can choose a nearby locker as their delivery location. This reduces the fragmentation of deliveries in the last mile, thereby helping to reduce the congestion and environmental pollution caused by urban freight trips (Morganti *et al.*, 2014), as well as reducing routing costs. In recent years, a new concept called *trunk/in-car delivery* has been proposed. Here, customers' packages can be delivered to the trunks of cars. Volvo launched its world-first in-car delivery service in Sweden in 2016 (Kirsten, 2016). The courier has a one-time digital code to get access to the trunk of the car. In April 2018, Amazon launched the in-car service in partnership with two major automakers General Motors and Volvo. This service is available in 37 cities in the US (Hawkins, 2018). Trunk delivery differs from home/workplace delivery and pick-up points delivery since the car moves and can be in different locations dur-

ing different periods of time, e.g., parked at the workplace in the morning and at the commercial center in the afternoon. As a consequence, synchronization between the car and the courier is required to make the delivery.

All these delivery services can be combined, and instead of choosing one delivery location during the online purchase, the customer can propose a set of delivery locations with the associated time constraints. To deliver a package to a specific customer, the courier only needs to choose one of the locations provided by the customer.

In this thesis, we aim to model and develop efficient solution methods for routing problems that arise in the context of last mile delivery when multiple shipping options are proposed: home/workplace, pick-up points, and car trunk. The last mile delivery with multiple shipping options allows customers to choose multiple locations to receive their packages. This provides customers more flexibility considering their convenience. Moreover, it might increase the rate of successful first-time deliveries and decrease the delivery costs. For example, in the United Kingdom, the cost of failed deliveries is almost $1.1 billion for retailers and e-commerce companies in a $100 billion market (Honorato, 2016; Symonds, 2018). Offering more delivery options could be profitable (BringgTeam, 2019).

We study both the single-vehicle and multi-vehicle routing problems in this context, i.e., the Generalized Traveling Salesman Problem with Time Windows (GTSPTW) and the Generalized Vehicle Routing Problem with Time Windows (GVRPTW), in which there are clusters representing possible delivery locations associated with a customer. It can be easily seen that in the problems we study, it is not necessary to visit all the locations associated with a given customer, since the courier only needs to deliver the package to one of the locations that the customer provides.

In the following, we outline the thesis.

- In Chapter 1, we survey non-Hamiltonian routing problems whose main characteristic is that only a subset of vertices of the problem graph need to be visited. For each of these problems, we give its definition and present a compact mathematical formulation. We also provide a literature review and give some of its applications.

- In Chapter 2, we study the single-vehicle routing problem for last mile delivery with multiple shipping options, which is called the Generalized Traveling Salesman Problem with Time Windows (GTSPTW). We present four integer

linear programming formulations for the GTSPTW. The models differ in the way we define the arc variables and time variables: based on vertices or clusters. Dominance relations between the linear relaxations of these formulations are theoretically established. Computational results on linear relaxations show that on average formulation $\mathcal{F}1$ (arc variables are defined between pairs of vertices and time variables are defined for all the vertices) is the best, followed by formulation $\mathcal{F}2$ (arc variables are defined as in $\mathcal{F}1$ while only one time variable is defined for each cluster). However, when the GTSPTW is solved using the branch-and-bound scheme in CPLEX, on average formulation $\mathcal{F}2$ is the most efficient, followed by formulation $\mathcal{F}1$. Therefore, we recommend using formulations $\mathcal{F}1$ and $\mathcal{F}2$ for the solution of the GTSPTW. In addition, supervalid inequalities for formulations $\mathcal{F}1$ and $\mathcal{F}2$ are proposed, and we experimentally show how they strengthen these models.

- In Chapter 3, we develop a branch-and-cut algorithm for the GTSPTW. Several families of valid inequalities are proposed, which contain polynomial or exponential numbers of constraints. They are incorporated in a branch-and-cut framework through dedicated separation procedures. A high quality initial solution is constructed based on a heuristic and used as a warm start in the branch-and-cut algorithm. We test the algorithm on three groups of instances with different characteristics. The results clearly demonstrate the efficiency of the proposed branch-and-cut algorithm and the quality of formulation $\mathcal{F}2$. The proposed algorithm based on formulation $\mathcal{F}2$ can solve instances around 30 clusters within one hour of computation time.

- In Chapter 4, we study the multi-vehicle case, which is called the Generalized Vehicle Routing Problem with Time Windows (GVRPTW). We propose an integer linear programming formulation and a set covering formulation for the GVRPTW. Based on the set covering formulation, we develop a column generation based heuristic to solve the GVRPTW. It combines several components including construction heuristic, route optimization procedure, local search, and the generation of negative reduced cost routes. Experimental results on benchmark instances show that the proposed algorithm is very efficient, and high-quality solutions can be obtained within very short computation times for instances with up to 120 clusters.

- Finally, we give some conclusions that we draw from the studied problems and discuss some perspectives of future work.

Chapter 1

A survey on non-Hamiltonian routing problems

Contents

1.1 Introduction

Urban population grows with a constant trend, and nowadays half of the population lives in urban areas. In particular, almost 75% of the European population lived in urban areas in 2015. Higher shares were recorded in Latin America and the Caribbean (79.8%) and North America (81.6%).

In 2014, the Netherlands, Belgium, and United Kingdom were the most concentrated urban regions of the European Union (EU) since 44.1%, 34.6% and 27.5% respectively of their total area was classified as predominantly urban. On the other side, nordic EU member States, Ireland and eastern EU member States such as Hungary, Romania, Croatia, and Bulgaria account for at least 97.0 % of their total surface as rural regions.

Moreover, in 2012, 53% of all gross domestic product (GDP) was generated in urban zones. As a consequence, even if the area occupied by predominantly urban regions across the EU was generally quite small, in light of the population that lives in, it is easy to see the concentration of economic activities in these regions. Same situation happens around the world, with more than 80% of global GDP generated in cities (Wahba, 2019).

These activities produce economic growth, engender a wide range of problems among those we can name traffic congestion, and pollution (Kotzeva, 2016) and rise a number of challenges in the context of *city logistics*. The concept of *city logistics* was defined by Taniguchi *et al.* (2001) as "*the process for totally optimizing the logistics and transport activities by private companies in urban areas while considering the traffic environment, the traffic congestion and energy consumption within the framework of a market economy*". The reader interested in city logistics is refereed to Cattaruzza *et al.* (2017); Crainic *et al.* (2009); Taniguchi *et al.* (2001).

Among the logistics solutions proposed to deliver parcels in the context of city logistics, several consider multiple locations as delivery points for a specific parcel. For example, a customer can choose to be delivered at home or at a nearby pick-up point. Pick-up points may be shops that offer collection services or lockers installed by the delivery company. Both lockers and pick-up points allow the transfer of the parcel from the courier to the consignee without the simultaneous presence of both.

A recent technology proposed by Volvo introduces the possibility of making delivery directly into the trunk of the customer's car. Volvo launched its world-first in-car delivery service in Sweden in 2016. The courier has a one-time digital code to get access to the trunk of the car. Since the car moves along the day, the parcel can be delivered in different locations and requires the simultaneous presence of the car and courier.

Classical routing problems studied by scholars are defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ that represent the delivery network. $\mathcal{V} = \{0, \ldots, N\}$ is the set of vertices, and $\mathcal{A}$ is the set of arcs connecting pairs of vertices. Vertex 0 represents the depot where the fleet of vehicles is located, while nodes $\mathcal{V} \setminus \{0\}$ represent customers. Classical routing problems assume that all the customers, namely all the vertices in the graph, have to be visited once by a single vehicle (Toth & Vigo, 2014).

However, in the two aforementioned examples in the city logistics context, each customer is associated with several possible delivery locations. To deliver the parcel, only one of the possible delivery locations associated with the same customer has to be selected to make the delivery. Then, a solution of such routing problems visits only a subset of the vertices of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ representing the instance.

Routing problems that do not require to visit all the vertices of the graph are various and are not limited to the fact that only one vertex need to be visited among a fixed subset of vertices as in the aforementioned examples. When a profit is associated to the visit of a vertex, some problems consider to only visit a subset of the vertices in order to maximize the profit, or to take into account some resource constraints. Other problems also consider that if a vertex is visited, some other vertices (e.g., the ones close from this visited vertex) do not need to be visited anymore.

In all the problems mentioned, only a subset of vertices of the graph need to be visited. We refer to this class of problems as non-Hamiltonian routing problems (non-HRP) (Laporte & Martín, 2007). Laporte & Martín (2007) propose a related survey that is limited to the single-vehicle case of the non-HRP and covers research works up

to the year 2005. Fischetti *et al.* (2007) also give a non-exhaustive list of single-vehicle non-HRP including works up to the year 2002. The aim of this paper is to survey recent advances in both the single-vehicle and the multi-vehicle non-HRP.

The paper is organized as follows. In Section 1.2 we describe the organization of the survey, the choices we made, the notations and the convention we use. Section 1.3 is dedicated to the non-HRP where one vehicle is in charge of the delivery service, while Section 1.4 considers non-HRP where a fleet of several vehicles is available. Finally, Section 1.5 concludes the paper.

1.2 Survey organization and notations

1.2.1 Survey organization

The paper is organized in two main sections. The first deals with one vehicle problems while the second section deals with the multi-vehicle case. Each section starts with the introduction and the formulation of the classical related routing problem from which the hamiltonian requirement has been removed, namely the Traveling Salesman Problem (TSP) and the capacitated Vehicle Routing Problem (VRP).

This paper is not intended to provide a detailed survey on each non-HRP class mainly because surveys dedicated to some specific classes of non-HRP are already present in the literature and such work would simply end up in a repetition. The main objective of this paper is to provide to the reader a complete list of non-HRP and to redirect he/she towards dedicated recent research papers or towards dedicated surveys when these are available.

For each problem, first we provide a mathematical formulation. The purpose of the formulation is descriptive only. We provide compact formulations, namely that are polynomial in the number of variables and constraints. This implies that subtour elimination constraints are expressed in the Miller-Tucker-Zemlin (MTZ) form (Miller *et al.*, 1960). Then we review the literature on this problem. The articles are divided into two categories according to the solving methodology proposed in the paper, i.e., exact methods and heuristic methods.

We limit the scope of this literature review to routing problems where the service is accomplished on the nodes of the graph and not on the arcs. As a consequence, we do not include problems that belong to the family of arc routing problems.

1.2.2 Notations

All along the paper we take the convention of using uppercase letters in mathematical calligraphy to represents sets. Uppercase letters represent data while lowercase letters represent variables.

Routing problems may be defined on a digraph that represents the road network. This digraph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the vertex set and $\mathcal{A}$ is the arc set. A specific vertex 0 represents a depot. The cardinality of the vertex set $\mathcal{V}$ is $N + 1$. Vertices $1, \ldots, N$ represent locations that may require a visit or a service. There is a fleet $\mathcal{F}, |\mathcal{F}| = F$ of vehicles located at the depot 0.

With each arc $(i, j) \in \mathcal{A}$, we associate a traveling time T_{ij}, a traveling cost C_{ij} and possibly a distance D_{ij}. If not differently specified, we suppose that traveling times, traveling costs and distances are asymmetric ($T_{ij} \neq T_{ji}$, $C_{ij} \neq C_{ji}$, $D_{ij} \neq D_{ji}$ for some $i, j \in \mathcal{V}$) and satisfy the triangle inequality ($T_{ij} \leq T_{ik} + T_{kj}$, $C_{ij} \leq C_{ik} + C_{kj}$ and $D_{ik} + D_{kj} \leq D_{ij}$ for all $i, j, k \in \mathcal{V}$).

When an undirected graph is considered, it is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set, while $\mathcal{E}$ is the edge set. Then the problem is symmetric, i.e., $T_{ij} = T_{ji}$, $C_{ij} = C_{ji}$, $D_{ij} = D_{ji}$ for all $i, j \in \mathcal{V}$. For ease of notation we indicate both the vertex sets of a digraph and an undirected graph by $\mathcal{V}$, while we distinguish the arc set of a digraph $\mathcal{A}$ from the edge set of an undirected graph $\mathcal{E}$.

To present mathematical models, there are some common variables used for different problems. Based on a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, variables x_{ij}, $\forall (i, j) \in \mathcal{A}$ are binary variables that represent the arc selection, equal to 1 if and only if arc (i, j) is used in the solution; variables y_i, $\forall i \in \mathcal{V}$ are binary variables that represent vertex selection, equal to 1 if and only if vertex i is used in the solution; variables u_i, $i \in \mathcal{V} \setminus \{0\}$ are real non-negative variables; variables t_i, $i \in \mathcal{V}$ are real non-negative variables that represent the beginning time of service at a vertex.

If there are more than one vehicle available ($F > 1$) at the depot, we might add one more index corresponding to the vehicle in the variable definition, i.e., we use variables x_{ijf}, $\forall (i, j) \in \mathcal{A}, f \in \mathcal{F}$ and y_{if}, $\forall i \in \mathcal{V}, f \in \mathcal{F}$ for arc selection and vertex selection respectively to represent that a certain arc or vertex is used by a specific vehicle or not.

There are some additional notations. Given a subset $\mathcal{S}$ of the vertex set $\mathcal{V}$, $\mathcal{S} \subset \mathcal{V}, \mathcal{S} \neq \emptyset$, we define $\delta^+(\mathcal{S}) = \{(i, j) \in \mathcal{A} | i \in \mathcal{S}, j \in \mathcal{V} \setminus \mathcal{S}\}$ and $\delta^-(\mathcal{S}) = \{(i, j) \in \mathcal{A} | i \in \mathcal{V} \setminus \mathcal{S}, j \in \mathcal{S}\}$, namely the set of arcs exiting from and entering into $\mathcal{S}$. When $\mathcal{S} = \{i\}$,

we use the notation $\delta^+(i)$ and $\delta^-(i)$ instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$ respectively. Last, we define $\gamma(\mathcal{S}) = \{(i,j) \in \mathcal{A} | i,j \in \mathcal{S}\}$ as the set of the arcs with both endpoints in $\mathcal{S}$.

1.3 One-vehicle case

This first part of the paper focuses on non-HRP where a single vehicle or person is in charge of the service operation. Usually when only one resource is available to perform operations, we refer to it as a *traveling salesman* rather than a *vehicle*. We begin the section by introducing the Traveling Salesman Problem (TSP). Then, we present different families of non-HRP derived from the TSP in which the Hamiltonian requirement is removed. These problems include the Generalized Traveling Salesman Problem, TSP with Profits, the Covering Tour Problem, the Covering Salesman Problem, the Median Cycle Problem, the Ring Star Problem, and the Traveling Purchaser Problem.

1.3.1 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the vertex set $\mathcal{V}$ represents the set of cities to be visited by the salesman. The TSP consists in determining the minimum cost Hamiltonian cycle on $\mathcal{G}$.

The TSP can be formulated using variables x_{ij} and u_i introduced in Section 1.2.2 as follows (Miller *et al.* (1960)):

$$(TSP) \quad \min \sum_{(i,j)\in\mathcal{A}} C_{ij}x_{ij} \tag{1.1}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = 1 \qquad \forall i \in \mathcal{V}, \tag{1.2}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = 1 \qquad \forall i \in \mathcal{V}, \tag{1.3}$$

$$u_i - u_j + Nx_{ij} \leq N - 1 \qquad \forall i,j \in \mathcal{V}\setminus\{0\}, i \neq j, \tag{1.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \mathcal{V}, i \neq j, \tag{1.5}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V}\setminus\{0\}. \tag{1.6}$$

The objective function (1.1) minimizes the cost of the Hamiltonian cycle. Constraints (1.2) and Constraints (1.3) impose that each vertex is visited exactly once. Constraints (1.4) are subtour elimination constraints in the Miller-Tucker-Zemlin (MTZ)

form. Constraints (1.5) and (1.6) define the variables. The TSP is one of the most studied combinatorial optimization problems. The interested reader is refereed to Applegate *et al.* (2006); Cook (2011); Gutin & Punnen (2007).

1.3.2 The Generalized Traveling Salesman Problem

The Generalized TSP (GTSP) is an extension of the TSP where the vertices of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ are partitioned into clusters, i.e., $\mathcal{C}_0 = \{0\}, \mathcal{C}_1, ..., \mathcal{C}_K$ clusters. $\mathcal{C}_0 \bigcup ... \bigcup \mathcal{C}_K = \mathcal{V}$ and $\mathcal{C}_h \cap \mathcal{C}_k = \emptyset, \forall h, k \in \mathcal{K}, h \neq k$, where $\mathcal{K} = \{0, 1, ..., K\}$ denotes the cluster index set. The arc set $\mathcal{A}$ contains arcs that link vertices belonging to different clusters, that is, $\mathcal{A} = \{(i,j) | i \in \mathcal{C}_k, j \in \mathcal{C}_l, k \neq l, k, l \in \mathcal{K}\}$. The objective of the GTSP is to find a minimum cost tour that visits each cluster exactly once.

The GTSP is formulated using variables x_{ij}, y_i and u_i introduced in Section 1.2.2 as follows:

$$(GTSP) \quad \min \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij} \tag{1.7}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.8}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.9}$$

$$\sum_{i\in\mathcal{C}_k} y_i = 1 \qquad \forall k \in \mathcal{K}, \tag{1.10}$$

$$u_i - u_j + K x_{ij} \leq K - 1 \qquad \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j, \tag{1.12}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{V}, \tag{1.13}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.14}$$

The objective function (1.7) minimizes the traveling cost of the tour. Constraints (1.8) and Constraints (1.9) are flow conservation constraints and impose that an arc enters and exits each selected vertex. Constraints (1.10) impose that exactly one vertex is visited per cluster. Constraints (1.11) are subtour elimination constraints in the MTZ form. Constraints (1.12) – (1.14) define the variables.

As in the TSP formulation (1.1)–(1.6), x_{ij} variables select the arcs used by the salesman to perform the tour. With respect to the TSP formulation (1.1)–(1.6), vertex

selection for each cluster is required. As a consequence, variables y_i are introduced to select the vertex that is visited for each cluster.

If all clusters $\mathcal{C}_i$, $i = 1, \ldots, K$ are singletons, the GTSP reduces to the TSP. From this observation it immediately follows, by reduction to the TSP, that the GTSP is NP-hard.

Polyhedral results

The paper by Fischetti *et al.* (1995) is dedicated to the description of the polytope of the symmetric GTSP ($T_{ij} = T_{ji}$, $C_{ij} = C_{ji}$). They prove that the dimension of the polytope of the GTSP is $|\mathcal{E}| - K$. Some families of inequalities are proved to be facet defining. Moreover they provide a general lifting procedure that allows to extend facet defining inequalities of the symmetric TSP polytope to a facet of the symmetric GTSP.

Transformation to the TSP

Several papers propose to solve the GTSP by transforming an instance of the GTSP into an instance of the well-studied TSP, and then solve the latter by applying existing exact or heuristic approaches for the TSP.

Noon & Bean (1993) propose a technique to transform any instance of the GTSP to an instance of the asymmetric TSP. This technique first constructs an instance of the clustered-TSP, as the GTSP admits a partition of the nodes in $\mathcal{V}$ into clusters. However the clustered-TSP requires to visit all the nodes of $\mathcal{V}$ with the constraints that the nodes in the same cluster must be visited consecutively. It introduces zero-cost arcs to form a cycle among nodes inside each cluster. Then the clustered-TSP is transformed into a standard asymmetric TSP by simply adding a large cost to all the inter-cluster arcs. The proposed transformation creates an instance of the TSP with $|\mathcal{V}|$ nodes and $|\mathcal{A}| + \sum_{k \in \mathcal{K}, |\mathcal{C}_k| > 1} |\mathcal{C}_k|$ arcs.

Dimitrijević & Šarić (1997) propose a transformation of an instance of the GTSP to an instance of the asymmetric TSP based on the replication of all the nodes in $\mathcal{V}$. They connect all the nodes in the same cluster via a zero-cost cycle. Similarly, the replication of nodes in the same cluster are connected via a zero-cost cycle. Arcs that start from the original node and end at the replication node are introduced and associated with a large cost. The resulting graph has $2|\mathcal{V}|$ nodes and $|\mathcal{A}| + |\mathcal{V}| + 2\sum_{k \in \mathcal{K}, |\mathcal{C}_k| > 1} |\mathcal{C}_k|$ arcs.

Laporte & Semet (1999) propose a transformation of an instance of the symmetric GTSP into the symmetric TSP made on the replication of all the nodes in $\mathcal{V}$. Nodes in the same cluster are arbitrarily ordered as $1, 1', 2, 2', \ldots, h, h'$, placing a vertex next to its copy. Edges $\{1, 1'\}, \{1', 2\}, \{2, 2'\}, \ldots, \{h, h'\}, \{h', 1\}$ are added. Edges connecting copies of the same node has a cost of $-M$, while other edges a cost of $-2M$, where M is a large enough value. The resulting graph has $2|\mathcal{V}|$ nodes and $(|\mathcal{E}| + 2\sum_{k=1}^{K}(|\mathcal{C}_k|))$ edges.

Ben-Arieh *et al.* (2003) and Zia *et al.* (2018) discuss "non-exact" transformations from the GTSP to TSP that provide heuristic solutions for the GTSP using existing algorithms for the TSP.

Exact methods

The existing literature on exact methods for the GTSP is quite limited. The GTSP is first introduced by Srivastava *et al.* (1969) and Saskena (1970a), and dynamic programming is used to solve it. Laporte & Nobert (1983) propose an integer programming formulation and a branch-and-bound approach for the symmetric GTSP. The largest instance solved contains 50 nodes and 10 clusters. Laporte *et al.* (1987) study the asymmetric case of the GTSP and propose a branch-and-bound algorithm. Instances with up to 100 nodes and 8 clusters are solved to optimality. Noon & Bean (1991) present a branch-and-bound approach for the asymmetric GTSP. They propose a Lagrangian relaxation to compute a lower bound and a heuristic to compute an upper bound. Non-optimal arcs and nodes are identified and eliminated based on the reduced costs. This method is tested on a set of randomly generated instances. They solve instances with up to 104 nodes and 8 clusters. Meanwhile, their algorithm is more efficient compared to the approach proposed by Laporte *et al.* (1987). Fischetti *et al.* (1997) study the symmetric GTSP. The authors propose an efficient branch-and-cut algorithm to solve the GTSP. They develop exact and heuristic separation procedures for some classes of facet-defining inequalities. They also generated a library of GTSP instances called GTSP-LIB by taking TSP-LIB instances and performing a clustering procedure on the nodes. Their algorithm could solve instances with up to 89 clusters and 442 nodes.

Heuristic methods

A different approach to tackle the GTSP is to develop heuristics. Heuristics include genetic algorithms (Ardalan *et al.*, 2015; Snyder & Daskin, 2006), particle swarm based approach (Shi *et al.*, 2007), and ant colony algorithms (Pintea *et al.*, 2007; Yang *et al.*, 2008). Several memetic algorithms combining genetic and powerful local search algorithms have also been proposed (Bontoux *et al.*, 2010; Gutin & Karapetyan, 2010). The memetic algorithm proposed by Gutin & Karapetyan (2010), called GK, obtains excellent results on the GTSP-LIB instances, with computation times shorter than 1 minute and most of the solutions within 0.2% of the best-known values. It has been shown in Drexl (2014) that GK performed well for instances with up to about 200 clusters. However, for instances with more than 500 clusters, the gaps to the optimal solutions usually exceeded 10%. Helsgaun (2015) extends the Lin-Kernighan-Helsgaun (LKH) TSP solver (Helsgaun, 2000, 2009) to the GTSP, called GLKH. He transforms the GTSP instance to the standard asymmetric TSP using the transformation method proposed by Noon & Bean (1993) and then solves the TSP using the LKH solver. The resulting algorithm improves the solution quality on GTSP-LIB instances compared with the GK proposed by Gutin & Karapetyan (2010), at the expense of more computation time. The GLKH is also tested on several other problem libraries. It could find high-quality solutions for new generated large-scale GTSP instances with up to 17180 clusters and 85900 vertices. It also shows strong performances on transformed instances of the arc routing problems (Corberán *et al.*, 2012). Smith & Imeson (2017) presented a solver called GLNS based on adaptive large neighborhood search. Their results show that on the benchmark GTSP-LIB instances, the GLNS shows similar performance to that of GK and GLKH. On several other problem libraries, given the same amount of computation time, the GLNS finds higher quality solutions than existing approaches.

Application

Following Laporte *et al.* (1987), one application of the GTSP is proposed in Bovet (1983) and consists in determining the locations of mailboxes in two phases. First a set of possible location of each mailbox is built. Thus, each cluster contains the potential locations of a mailbox. The exact locations of mailboxes are determined while calculating the route of the postal van that consists of a single visit of each cluster such that routing cost is minimized.

Saskena (1970b) describe an application where a person needs to receive different services and each service is provided by different agencies. Agencies are then clustered with respect to the service they provide. The tour the clients perform consist in visiting each cluster once while minimizing routing costs.

Laporte *et al.* (1996) mention the material flow system design problem. A production plant is partitioned in several production zones. These zones are supposed to be polygonal. The problem consists in designing a minimal length tour such that it contains at least one vertex of each zone. Then, each cluster contains the vertices of the zones and the problem can be modeled as a GTSP.

1.3.3 The TSP with Profits

The TSP with Profits (TSPPs) is a generalization of the TSP where in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ each vertex $i \in \mathcal{V} \setminus \{0\}$ is associated with a profit P_i. The objective is to optimize both the collected profit and the traveling cost. As such, the visit of all the vertices is not mandatory. The collection of the profits and the traveling cost optimization may be considered in the objective function or in the constraints. In particular, based on the different case the TSPPs can be classified as follows:

- The Profitable Tour Problem (PTP) (Dell'Amico *et al.*, 1995) where the objective is to minimize the traveling cost minus the collected profit.
- The Orienteering Problem (OP) (Chao *et al.*, 1996a) where the objective is to find a tour that maximizes the profit under a constraint that imposes the maximum cost C_{max} of the tour.
- The Prize-Collecting Traveling Salesman Problem (PCTSP) (Balas, 1989) where the objective is to minimize the tour traveling cost under a constraint that imposes a minimum prize collection P_{min}.

The TSPPs can be formulated using variables x_{ij}, y_i and u_i introduced in Section 1.2.2 as follows:

$$(TSPPs) \quad \min f(x, y) \tag{1.15}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.16}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.17}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.18}$$

$$g(x, y) \geq 0, \tag{1.19}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j, \tag{1.20}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{V}, \tag{1.21}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.22}$$

where $f(x, y)$ and $g(x, y)$ are as follows.

- for the PTP:
  - $f(x, y) = \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij} - \sum_{i\in\mathcal{V}} P_i y_i$
  - $g(x, y) = 0$;
- for the OP:
  - $f(x, y) = -\sum_{i\in\mathcal{V}} P_i y_i$
  - $g(x, y) = C_{max} - \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij}$;
- for the PCTSP:
  - $f(x, y) = \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij}$
  - $g(x, y) = \sum_{i\in\mathcal{V}} P_i y_i - P_{min}$.

The objective function (1.15) is to minimize the traveling cost minus the collected profit in the case of the PTP, is to maximize the profit in the case of the OP and is to minimize the traveling cost in the case of the PCTSP. Constraints (1.16) and Constraints (1.17) are flow conservation constraints and impose that an arc enters and exits each selected vertex. Constraints (1.18) are MTZ subtour elimination constraints. Constraints (1.19) are inactive for the PTP, impose the maximum tour cost for the OP and the minimum prize to collect for the PCTSP. Constraints (1.20) – (1.22) define the variables.

The three versions of the TSPPs are NP-hard (Feillet *et al.* (2005)).

The interested reader is referred to the surveys by Feillet *et al.* (2005), Archetti *et al.* (2014), Vansteenwegen *et al.* (2011) and Gunawan *et al.* (2016). Feillet *et al.* (2005) reports polyhedral results on the TSPPs and its transformation into the TSP and covers

works up to the year 2000. Archetti *et al.* (2014) is dedicated to the Vehicle Routing Problem with Profits, but includes research works dedicated to both single-vehicle case (TSPPs) as well as the multi-vehicle case, up to the year 2014. Vansteenwegen *et al.* (2011) provide a comprehensive survey on the OP and its variants, including problem descriptions, benchmark instances and solution approaches, covering works up to the year 2009. Gunawan *et al.* (2016) also surveys on the OP extending new works and new variants of the OP up to the year 2015.

In this section, in order to avoid the repetition, we focus on the related literature on the PTP and PCTSP published since 2014 and on the OP since 2015, as well as on papers that were not reported. Recent works on the PTP and its variants include both exact and heuristic approaches, while all the recent works on the PCTSP, OP and their variants propose heuristic approaches.

Exact methods: PTP

To the best of our knowledge, there is no specifically proposed exact approach for the PTP. However, there exist some recent works on variants of the PTP, e.g., the capacitated PTP (Jepsen *et al.*, 2014), the time-dependent PTP (Lera-Romero & Miranda-Bront, 2019; Sun *et al.*, 2018), etc., as described in the following. Jepsen *et al.* (2014) propose a branch-and-cut algorithm for the capacitated PTP. In the capacitated PTP, each customer is associated with a profit and a demand, and a capacity is given for the maximum load of the tour. The objective is to find a tour that minimizes the total traveling cost minus the profits gained from the visited customers, with the demand accumulated at the customers does not exceed the capacity. Sun *et al.* (2018) study the time-dependent capacitated PTP with time windows and precedence constraints. In this problem, a single vehicle with capacity limit is available. To deal with road congestion, traveling times are considered to be time-dependent. They propose an exact approach by developing a tailored labeling algorithm. A heuristic is also described which could obtain high-quality solution in lower computation time than the proposed labeling algorithm. Lera-Romero & Miranda-Bront (2019) study the time-dependent PTP with resource constraints. These constraints can be related to time windows, vehicle capacity, duration of the route, etc. The authors propose a mixed integer programming formulation and four new families of valid inequalities for the problem. They develop a branch-and-cut algorithm, and experimental results on four different problems show that the proposed approach is effective and flexible.

Heuristic methods: PTP

There exist two heuristic works on variants of the PTP. Lee *et al.* (2010) propose a genetic algorithm for the PTP with pickup and delivery. Chentli *et al.* (2018) propose an adaptive large neighborhood search for the PTP with simultaneous pickup and delivery.

Heuristic methods: PCTSP

Archetti *et al.* (2014) cover most of the exact, heuristic, and approximation algorithms for PCTSP up to the year 2014. In recent years, only a few works appeared on the PCTSP. Pedro *et al.* (2013) propose a tabu search approach for the asymmetric version of the PCTSP. da Silva Menezes *et al.* (2014) study the prize-collecting traveling car renter problem. In the traveling car renter problem, several cars are available to be used during the tour and a tourist wants to visit a set of cities with rented cars. The objective is to determine a minimum cost tour that visits some cities with different rented vehicles, at least reaching a pre-specified satisfaction. The authors proposed a memetic algorithm to solve this problem.

Heuristic methods: OP

For the OP, there are several recent works. Kara *et al.* (2016) present two polynomial-size formulations. Kobeaga *et al.* (2018) propose a population-based evolutionary algorithm for the OP, whose main characteristic is to maintain unfeasible solutions during the search and to use specific operators to recover feasibility when it is required. Experimental results show that the proposed algorithm is competitive for medium-size instances with up to 400 nodes and is excellent for large-size instances with up to 7397 nodes in terms of quality and time. Santini (2019) proposes an adaptive large neighborhood search algorithm for the OP. Computational results showed that it is competitive with the genetic algorithm proposed by Kobeaga *et al.* (2018) and it finds better solutions when given a long CPU time limit. Moreover, the two algorithms seem to be complementary in the sense that for the sets of large instances whose best solution is found by genetic algorithm and the proposed algorithm are disjoint.

The rest of recent papers work on the variants of the OP, e.g., OP with mandatory visits and exclusionary constraints, multi-objective OP, time-dependent OP, stochastic OP, probability OP, the set OP. These variants are reviewed hereafter.

Palomo-Martínez *et al.* (2017) study a variant of the OP in which mandatory visits for certain nodes and incompatibility between nodes are considered. They propose a hybrid variable neighborhood search algorithm combining the greedy randomized adaptive search procedure. Lu *et al.* (2018) propose a memetic algorithm for the same problem. Experimental results on the benchmark instances proposed by Palomo-Martínez *et al.* (2017) show that their algorithm is highly effective and outperforms the hybrid variable neighborhood search algorithm.

Martín-Moreno & Vega-Rodríguez (2018) propose an evolutionary algorithm for the bi-objective OP. Experimental results show that the proposed algorithm outperforms the other two state-of-the-art algorithms for the bi-objective OP.

Mei *et al.* (2016) study the multi-objective time-dependent OP, in which the time-dependent traveling time and multiple preferences are taken into account. They propose two metaheuristics, i.e., a multi-objective memetic algorithm and a multi-objective ant colony algorithm to solve the problem.

Verbeeck *et al.* (2016) introduce the OP with time windows and time-dependent stochastic traveling time. They design an ant colony algorithm to solve the problem. In this variant, the traveling time between two locations is a stochastic function that depends on the departure time at the first location.

Angelelli *et al.* (2017) study a variant of the OP called the probabilistic OP, in which each node will be available for visit only with a certain probability. The authors formulate the problem as a stochastic mixed integer programming problem and propose a branch-and-cut approach and several metaheuristics. Computational results prove the efficiency of the exact method, and the metaheuristics can find high quality solutions in a few minutes.

Varakantham *et al.* (2018) study a variant of the stochastic OP, in which the traveling time distribution for moving from one vertex to another depends on the arrival time at the former vertex.

Dolinskaya *et al.* (2018) model the search and rescue operation in a post-disaster as a variant of the OP, in which multiple paths with stochastic traveling times exist between nodes.

Bian & Liu (2018) focus on the operational-level stochastic OP, in which the vehicle can adjust the routing plan in real-time.

Freeman *et al.* (2018) study a variant of the OP called attractive OP for planning entertainment events, specifically, the concert touring industry. The problem seeks to

determine a maximum profit tour and event plan among a set of candidate locations over a fixed time horizon. The profit from a particular event depends on attendance.

Pěnička *et al.* (2019) study the set OP. In the set OP, the customers are grouped into clusters, and the profit associated with each cluster is collected by visiting at least one of the customers in the respective cluster. The authors propose a variable neighborhood search for this problem. Computational results show that the proposed algorithm improves the solutions of SOP benchmark instances in significantly less computation time than the existing approaches.

Yu *et al.* (2019) study the OP with service time dependent profits. In this variant, the profit collected at each node is a non-linear function of service time. A metaheuristic is proposed to decompose the problem into a routing subproblem and a scheduling subproblem.

Applications

The OP is an important problem that has several real-world applications. The most famous and the most studied is the tourist trip design problem. The interested reader is referred to the survey on solving tourist trip design problem by Gavalas *et al.* (2014). Tourists visiting a destination for one or several days must decide which points of interest (POIs) would be more interesting to visit and to determine a route for each trip day, i.e., which POIs to visit as well as the visiting order among them. The objective is to maximize tourist satisfaction (profit) while respecting constraints such as opening hours of POIs, the traveling distances between POIs, the daily time available for sightseeing.

1.3.4 The Covering Tour Problem and the Covering Salesman Problem

Gendreau *et al.* (1997) introduce the Covering Tour Problem (CTP). It is defined on a graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{W}, \mathcal{A})$, where $\mathcal{V} \cup \mathcal{W}$ is the vertex set, $\mathcal{V}$ is a set of vertices that can be visited, $\mathcal{T} \subseteq \mathcal{V}, 0 \in \mathcal{T}$ is a set of vertices that must be visited and $\mathcal{W}$ is a set of vertices that must be covered, i.e., that must lie within a prespecified distance D from a visited vertex on the tour. The CTP consists of determining a minimum length tour over a subset of $\mathcal{V}$ in such a way that all vertices in $\mathcal{T}$ are visited, and every vertex in $\mathcal{W}$ is covered.

We say that a vertex j is covered by vertex i if $D_{ij} \leq D$. Let $\mathcal{S}_j = \{i \in \mathcal{V} | D_{ij} \leq D\}$ be the set of vertices that cover vertex $j \in \mathcal{W}$.

The CTP can be formulated as follows:

$$(CTP) \quad \min \sum_{(i,j)\in\mathcal{A}} D_{ij}x_{ij} \tag{1.23}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.24}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.25}$$

$$\sum_{i\in\mathcal{S}_j} y_i \geq 1 \qquad \forall j \in \mathcal{W}, \tag{1.26}$$

$$y_i = 1 \qquad \forall i \in \mathcal{T}, \tag{1.27}$$

$$u_i - u_j + Nx_{ij} \leq N - 1 \qquad \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.28}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j, \tag{1.29}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{V}, \tag{1.30}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.31}$$

The objective function (1.23) is to minimize the tour length. Constraints (1.24) and Constraints (1.25) are flow conservation constraints and impose that an arc enters and exits each selected vertex. Constraints (1.26) make sure that every vertex in $\mathcal{W}$ is covered by the tour. Constraints (1.27) guarantee that every vertex in $\mathcal{T}$ is visited by the tour. Constraints (1.28) are MTZ subtour elimination constraints. Constraints (1.29) – (1.31) define the variables.

The CTP is NP-Hard as it reduces to the TSP when $\mathcal{V} = \mathcal{W}$, $\mathcal{T} = \mathcal{V}$ and $D = 0$, $\forall i \in \mathcal{V}$.

The CTP can be formulated as a GTSP. We define for each $j \in \mathcal{W}$, the set $\mathcal{S}_j = \{i \in \mathcal{V} | D_{ij} \leq D\}$ as a cluster and for each $i \in \mathcal{T}$, the set $\mathcal{S}_i = \{i\}$ as a cluster. Note that if a vertex is in several clusters, then we replicate it as many time as the clusters it belongs to, in order to have a partition. Then solve a GTSP on these clusters solves the CTP.

A special case of the CTP is obtained when $\mathcal{W} = \mathcal{V}$, i.e., all the vertices that must be covered. The resulting problem is usually called the Covering Salesman Problem (CSP) and was introduced by Current & Schilling (1989). Each vertex $i \in \mathcal{V}$ is associated with a covering distance D_i. We say that a vertex j is covered by vertex i if $D_{ij} \leq D_i$.

For all vertices $j \in \mathcal{V}$ we define $\mathcal{S}_j = \{i \in \mathcal{V} | D_{ij} \leq D_i\}$. If vertex i is visited, then it is covered by itself. To obtain a formulation for the CSP, in the model (1.23)–(1.31) we remove Constraints (1.27) and replace Constraints (1.26) with

$$\sum_{i \in \mathcal{S}_j} y_i \geq 1, \forall j \in \mathcal{V}. \tag{1.32}$$

which means that all the vertices are visited or covered.

The CSP is NP-hard since when $D_i < \min_{j \in \mathcal{V}, j \neq i} D_{ij}$ for all $i \in \mathcal{V}$ it reduces to the TSP.

Collections of works on the CTP can be found in Fischetti *et al.* (2007) and Laporte & Martín (2007). In the next sections, we report all the works that have been done for the CTP and CSP.

Polyhedral results

Gendreau *et al.* (1997) propose several classes of valid inequalities for the CTP. They investigate the polyhedral properties of a family of constraints and some constraints are proved to be facet-defining.

Exact methods

The only exact method proposed for the CTP is a branch-and-cut algorithm presented by Gendreau *et al.* (1997). The authors provide several valid inequalities for the CTP. Their algorithm can solve randomly generated instances with up to 100 nodes.

Ozbaygin *et al.* (2016) study a variant of the CSP called the time constrained maximal CSP. In this problem, every vertex is associated with a demand and the objective is to maximize the amount of demand covered visiting a subset of vertices within a limited time. The authors propose two formulations and valid inequalities for this problem. A branch-and-cut algorithm is developed to solve this problem.

Heuristic methods

Concerning the CTP, Baldacci *et al.* (2005) propose a two-commodity flow formulation and three scatter search methods for the CTP. Kubik (2007) proposes several heuristics including ant colony algorithm for the CTP. Murakami (2018a) deals with the large-scale CTP which contains tens of thousands of vertices. They propose a heuristic based

on ruin and recreate. Computational results show that their algorithm outperforms the existing methods.

Some variants and generalizations of the CTP have also been studied in the literature. Motta *et al.* (2001) propose a greedy randomized adaptive search procedure for a generalized CTP in which the vertices in $\mathcal{W}$ can also be visited. Jozefowiez *et al.* (2007) study a bi-objective CTP where the two objectives are to minimize the tour length and to minimize the greatest distance between the covered node and its nearest visited node. The authors propose a multi-objective evolutionary algorithm to solve it.

Concerning the CSP, Salari & Naji-Azimi (2012) propose an integer linear programming based heuristic. Salari *et al.* (2015) give a polynomial-size formulation for the CSP and describe a hybrid heuristic algorithm combining ant colony optimization and dynamic programming technique. Computational results indicate the efficiency of the algorithm, especially for large size instances with more than 500 vertices.

Some generalizations of the CSP have also been studied in the literature. Current & Schilling (1994) study two multi-objective variants of the CSP, i.e., the median tour problem (MTP) and the maximal covering tour problem (MCTP). In both problems the tour must visit a predetermined number of nodes. Each node is associated with a demand. The first objective for both problems is to minimize the tour length. For the MTP, the second objective is to minimize the total distance between each unvisited node and the nearest visited node. For the MCTP, the second objective is to maximize the total demand that is covered within some prespecified maximal travel distance from a visited node.

Golden *et al.* (2012) study a variant of the CSP, in which nodes are associated with an additional visiting cost and a demand that represents the minimum number of times the node has to be covered. The objective is to minimize the total cost, which is the sum of the tour length and the fixed costs associated with the visited nodes. The authors develop two local search heuristics to solve this problem.

Another generalization of the CSP called the generalized covering TSP is studied in Shaelaie *et al.* (2014) and Pandiri & Singh (2019). The objective of this problem is to find a minimum length tour passing through a subset of facilities while covering at least a predetermined number of customers. Shaelaie *et al.* (2014) propose node-based and flow-based formulations for the problem, as well as two metaheuristic approaches, i.e., the memetic algorithm and the variable neighborhood search algorithm. Pandiri & Singh (2019) propose an artificial bee colony algorithm for this problem, and the

experimental results show the efficiency of their approach compared with heuristics proposed by Shaelaie *et al.* (2014).

The time constrained maximal CSP is studied in Naji-Azimi & Salari (2014). In this problem, a set of vertices is given including a depot, customer and facility vertices and the objective is to maximize the total number of covered customers by a tour over a subset of facilities within a given time limit. Node-based and flow-based mathematical models and a heuristic method are proposed for this problem.

Application

There are various applications of the CTP or CSP. For example, the construction of routes for visiting health care teams in developing countries can be modeled as the CTP. The health care team can only access a limited number of villages due to infrastructural restrictions, but all the people must be within a walking distance of the visited villages. The health care team's goal is to minimize traveling cost to see as many patients as possible (Current & Schilling, 1989; Hodgson *et al.*, 1998).

Another example is to determine the locations of post boxes among a set of candidates. The post boxes must be located within a reasonable distance from every household. Then, the aim of the post once is to minimize the cost of a collection route through all post boxes (Labbé & Laporte (1986)).

A similar example is to locate a number of regional distribution centers among a set of candidate sites in such a way that all customers are within a reasonable distance from at least one regional distribution center and that the cost of delivery and pick-up routes is minimized. This example is also applied to a decision of the location of satellite distribution centers to provide humanitarian supplies (Naji-Azimi *et al.*, 2012a).

1.3.5 The Median Cycle Problem and the Ring Star Problem

The Median Cycle Problem (MCP) is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, and a non-negative assignment cost A_{ij} is associated with each arc $(i, j) \in \mathcal{A}$. These costs may represent, for example, the amount to pay for serving location i from location j, or the distance between two vertices. Vertex 0 represents the depot and must be part of the solution. The MCP looks for a minimum cost tour that starts and ends at the depot

and goes through a subset of vertices $\mathcal{V}' \subseteq \mathcal{V}, 0 \in \mathcal{V}'$ such that the assignment cost of the tour does not exceed a given value A.

The assignment cost of a solution is defined as $\sum_{i \in \mathcal{V} \setminus \mathcal{V}'} \min_{j \in \mathcal{V}'} A_{ij}$, i.e., the sum of the assignment costs between each vertex not on the tour and its closest vertex on the tour.

Instead of using y_i variables to represent whether vertex $i \in \mathcal{V}$ is visited or not on the tour, variables $y_{ij}, \forall i, j \in \mathcal{V}$ are introduced. y_{ij} is a binary variable equal to 1 if and only if vertex i is assigned to vertex j on the tour. Notice that if a vertex i is on the tour, it is then assigned to itself, i.e., $y_{ii} = 1$. Then, variables y_{ij} are in charge of determining visits (using variables y_{ii} that play the role of variables y_i in the other problems) and assignments.

The MCP can be formulated as follows:

$$
\begin{aligned}
(MCP) \quad \min \quad & \sum_{(i,j) \in \mathcal{A}} C_{ij} x_{ij} && && (1.33)\\
\text{s.t.} \quad & \sum_{(i,j) \in \delta^+(i)} x_{ij} = y_{ii} && \forall i \in \mathcal{V}, && (1.34)\\
& \sum_{(j,i) \in \delta^-(i)} x_{ji} = y_{ii} && \forall i \in \mathcal{V}, && (1.35)\\
& \sum_{j \in \mathcal{V}} y_{ij} = 1 && \forall i \in \mathcal{V}, && (1.36)\\
& y_{ij} \leq y_{jj} && \forall i, j \in \mathcal{V}, && (1.37)\\
& u_i - u_j + N x_{ij} \leq N - 1 && \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, && (1.38)\\
& \sum_{(i,j) \in \mathcal{A}, i \neq j} A_{ij} y_{ij} \leq A, && && (1.39)\\
& x_{ij} \in \{0, 1\} && \forall i, j \in \mathcal{V}, i \neq j, && (1.40)\\
& y_{ij} \in \{0, 1\} && \forall i, j \in \mathcal{V}, && (1.41)\\
& u_i \geq 0 && \forall i \in \mathcal{V} \setminus \{0\}. && (1.42)
\end{aligned}
$$

The objective function (1.33) is to minimize the traveling cost. Constraints (1.34) and Constraints (1.35) are flow conservation constraints imposing that an arc enters and exits each visited vertex. Constraints (1.36) make sure that every vertex is on the tour or assigned to a vertex on the tour. Constraints (1.37) impose that a vertex can only be assigned to a visited vertex. Constraints (1.38) are MTZ subtour elimination

constraints. Constraints (1.39) guarantee that the assignment cost of the tour does not exceed the given value A. Constraints (1.40) – (1.42) define the variables.

The Ring Star Problem (RSP) is the problem where the assignment cost is minimized in the objective function instead of being taken into account in a constraint. Thus, the RSP consists in determining a tour (called as well ring) that minimizes the sum of the traveling and the assignment costs. The name *ring* comes from the application of the problem in the telecommunication network design (Labbé *et al.*, 2004). A model for the RSP is obtained from the model (1.33)–(1.42) for the MCP by replacing the objective function (1.33) by

$$(RSP) \quad \min \Big(\sum_{(i,j)\in\mathcal{A}} C_{ij}x_{ij} + \sum_{(i,j)\in\mathcal{A}, i\neq j} A_{ij}y_{ij} \Big) \tag{1.43}$$

and by removing Constraint (1.39).

The problem is NP-hard since the special case in which the assignment costs are very large compared to the traveling costs reduces to the TSP.

Note that the CTP and MCP/RSP are all problems related to the concept of coverage. In comparing the RSP with CTP, the RSP considers assignment costs in the objective function, and a nonvisited vertex is assigned to a single visited vertex, while the CTP takes into account only the traveling costs and a nonvisited vertex is covered by at least one visited vertex. When comparing the MCP with CTP, both problems only consider the traveling costs in the objective function, but the MCP has an upper bound for the assignment costs.

In the following, we survey all the works that have been done for the MCP and RSP.

Polyhedral results

Labbé *et al.* (2004) propose several classes of valid inequalities for the RSP. Dimension and facet-defining results are derived for the RSP. Kedad-Sidhoum & Nguyen (2010) propose a new formulation for the RSP. New facet-defining inequalities are derived and can improve the linear relaxation.

Exact methods

Labbé *et al.* (1999, 2005) are the first to study the MCP. They propose valid inequalities

and embed them within a branch-and-cut algorithm through separation procedure.

Labbé *et al.* (2004) introduce the RSP when studying a generic telecommunication network. They provide a mixed integer linear programming formulation and several classes of facet-defining inequalities for the RSP. A branch-and-cut algorithm is proposed which can solve instances with up to 300 nodes to optimality.

Kedad-Sidhoum & Nguyen (2010) propose a novel formulation and new facet defining inequalities for the RSP. They develop an efficient branch-and-cut algorithm whose results improve those proposed in Labbé *et al.* (2004).

Simonetti *et al.* (2011) model the RSP as a minimum Steiner arborescence problem. They develop a branch-and-cut algorithm, and a greedy randomized adaptive search procedure is used to determine good upper bounds. Experimental results show the superiority of the proposed method over the one by Labbé *et al.* (2004).

Heuristic methods

Pérez *et al.* (2003) propose a metaheuristic for the MCP combining variable neighborhood and tabu search. Renaud *et al.* (2004) propose two heuristics for both the MCP and RSP, i.e., the multistart greedy heuristic and a random keys evolutionary algorithm. Dias *et al.* (2006) propose a hybrid heuristic for the RSP based on variable neighborhood search and a greedy randomized adaptive search procedure, which most of the time performs better than the heuristic proposed by Pérez *et al.* (2003). Calvete *et al.* (2013) propose an evolutionary algorithm based on a new formulation of the RSP as a binary bi-level programming problem with one leader and two followers. The leader decides which vertices to include in the ring, one follower decides the connections of the tour, and the other follower decides about the assignment of the vertices not visited on the tour. Computational results show that the proposed algorithm outperforms the heuristics in the literature both in terms of the solution quality and computation time.

There are several works dealing with the bi-objective RSP. The objectives in these works are to minimize the ring cost and to minimize the assignment cost simultaneously (Calvete *et al.*, 2016; Liefooghe *et al.*, 2008a,b,c, 2010). Liefooghe *et al.* (2010) provide a set of four population-based metaheuristics to approximate the efficient set for the bi-objective RSP. Then, the authors propose two cooperative schemes between the two algorithms. Computational results show the effectiveness of the hybrid approaches, especially in large size instances up to 1002 nodes. Calvete *et al.* (2016) pro-

pose a hybrid metaheuristic which embeds a local search procedure in a multi-objective evolutionary algorithm to approach the Pareto front. They use a new chromosome encoding method, i.e., the chromosome does not provide the ring, but the nodes in the ring.

Applications

The RSP arises in the telecommunications networks design (Labbé *et al.*, 2004). The goal is to connect terminals to concentrators by point-to-point links, resulting in a star topology, and the concentrators are interconnected through a ring structure. The problem consists in selecting a subset of user locations where concentrators will be installed and interconnected by a ring network, and the other user locations are assigned to those concentrators. The objective is to minimize the total cost of all connections. The RSP also models logistic problems where the retailers in the ring are served by a single vehicle and are used as small depots from which the remaining retailers are supplied (Calvete *et al.*, 2013).

1.3.6 The Traveling Purchaser Problem

The Traveling Purchaser Problem (TPP) is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ includes the depot 0 and a set of markets that offer a set of products $\mathcal{Q}$. For each market $i \in \mathcal{V} \setminus \{0\}$ and for each product $q \in \mathcal{Q}$ a product availability Q_{iq} and a price P_{iq} are given. Moreover, a demand Q_q for each product $q \in \mathcal{Q}$ has to be satisfied. The TPP consists in determining a tour that visits a subset of the markets in order to buy enough products to satisfy the demand and to minimize traveling and purchasing costs. The interested reader is referred to Manerba *et al.* (2017) for a recent survey.

Variables $z_{iq}, \forall i \in \mathcal{V} \setminus \{0\}, q \in \mathcal{Q}$ are introduced to determine the quantity of a product q that is purchased at market i.

The TPP can be formulated using variables x_{ij}, y_i, u_i introduced in Section 1.2.2 and z_{iq} as follows:

$$(TPP) \quad \min \sum_{(i,j)\in\mathcal{A}} C_{ij}x_{ij} + \sum_{i\in\mathcal{V}\setminus\{0\}, q\in\mathcal{Q}} P_{iq}z_{iq} \tag{1.44}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V}, \tag{1.45}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i \qquad \forall i \in \mathcal{V}, \qquad (1.46)$$

$$\sum_{i\in V\setminus\{0\}} z_{iq} = Q_q \qquad \forall q \in \mathcal{Q}, \qquad (1.47)$$

$$z_{iq} \leq Q_{iq} y_i \qquad \forall i \in \mathcal{V}\setminus\{0\}, \qquad (1.48)$$

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad \forall i,j \in \mathcal{V}\setminus\{0\}, i \neq j, \qquad (1.49)$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \mathcal{V}, i \neq j, \qquad (1.50)$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{V}, \qquad (1.51)$$

$$z_{iq} \geq 0 \qquad \forall i \in \mathcal{V}\setminus\{0\}, q \in \mathcal{Q}, \qquad (1.52)$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V}\setminus\{0\}. \qquad (1.53)$$

The objective function (1.44) is to minimize the traveling and purchasing costs. Constraints (1.45) and Constraints (1.46) are flow conservation constraints and impose that an arc enters and exits each selected vertex. Constraints (1.47) ensure that the demands of all products are exactly satisfied. Constraints (1.48) impose that a market has to be visited to be able to supply a product and the quantity of a product purchased at a market is less than the available quantity at this market. Constraints (1.49) are MTZ subtour elimination constraints. Constraints (1.50) – (1.53) define the variables.

The TPP is NP-hard since it generalizes the TSP. The reader interested in polyhedral results on the TPP is referred to Manerba *et al.* (2017). Since the survey on the TPP and its variants by Manerba *et al.* (2017) covers works up to the year 2016, here we only summarize several works that have been published afterwards.

Exact methods

There are several works related to the variants of the TPP. A bi-objective TPP, called the green TPP, is proposed by Hamdan *et al.* (2017). One objective is to minimize the traveling and purchasing costs, and the other is to minimize the CO_2 emissions. The authors solve this problem using a branch-and-cut algorithm after transforming the model into a single objective formulation by the weighted comprehensive criterion method. Hamdan *et al.* (2018) propose a bi-objective integer linear programming model for what they called the sustainable TPP. They associate with each supplier a sustainability score and one of the objective is to maximize the total sustainability score of purchasing.

Heuristic methods

Bernardino & Paias (2018) present several metaheuristics combining genetic algorithms and local search to solve the uncapacitated TPP. In uncapacitated TPP, if a product is available in one market, then it is assumed that the quantity of this product is enough to fulfill the demand. Their metaheuristics can provide the best-known results for the high-dimensioned asymmetric instances, meanwhile provide better upper bounds for some symmetric instances with unknown optimal values. Skinderowicz (2018) propose an ant colony based algorithm for the uncapacitated TPP. Computational results show that it is competitive to the current state-of-the-art metaheuristic (Goldbarg *et al.*, 2009) for the uncapacitated TPP.

Palomo-Martínez & Salazar-Aguilar (2019) study a variant of the bi-objective TPP in which the purchased products must be delivered to a set of customers. The objectives are to minimize the total cost and to minimize the waiting time of the customers. The authors propose an efficient variable neighborhood search method.

Applications

The most common application of the TPP is in the procurement logistics, for example, in some companies' procurement operations. There is a commercial web application called "le bon côté des choses" (`https://www.leboncotedeschoses.fr/`) in which a purchaser selects his location, the list of products to purchase and a set of markets that he is willing to visit, then he can receive the most convenient shopping plan. Another application arises in the school bus routing (Riera-Ledesma & Salazar-González, 2012). The problem is to plan the tour for the school bus to pick up students from different stops. Here suppliers correspond to bus stops and products to students. More applications can be found in Manerba *et al.* (2017).

1.3.7 Discussion on the subtour elimination constraints

In the formulations proposed in the previous sections, subtour elimination constraints are expressed in the MTZ form. This form allows to provide formulations that have a polynomial number of variables and constraints with respect to the cardinality of $\mathcal{V}$. It is known that linear relaxations of such formulations usually provide poor bounds.

Tighter formulations can be obtained by replacing the subtour elimination constraints in the MTZ form by exponential families of constraints. This provides formulations that still have a polynomial number of variables with respect to the cardinality of $\mathcal{V}$, but are characterized by an exponential number of constraints. For the TSP, Constraints (1.4) are replaced by the so-called subtour elimination constraints expressed in the outer form:

$$\sum_{(i,j)\in\delta^+(\mathcal{S})} x_{ij} \geq 1, \quad \forall \mathcal{S} \subset \mathcal{V}, 2 \leq |\mathcal{S}| \leq |\mathcal{V}| - 2. \tag{1.54}$$

or by the equivalent inner form:

$$\sum_{(i,j)\in\gamma(\mathcal{S})} x_{ij} \leq |\mathcal{S}| - 1, \quad \forall \mathcal{S} \subset \mathcal{V}, 2 \leq |\mathcal{S}| \leq |\mathcal{V}| - 2. \tag{1.55}$$

Constraints (1.54) impose that at least one arc exiting each subset $\mathcal{S} \subsetneq \mathcal{V}$ is selected, while Constraints (1.55) impose that not more than $|\mathcal{S}| - 1$ arcs with both endpoints in $\mathcal{S}$ are selected.

In non-HRP, a subtour elimination constraint has to be imposed on a subset $\mathcal{S} \subseteq \mathcal{V}$ only if the subset is *visited*, namely only if at least one vertex in $\mathcal{S}$ is selected. As a consequence, Constraints (1.54) are not valid for non-HRP and need to be replaced by the following constraints

$$\sum_{(i,j)\in\delta^+(\mathcal{S})} x_{ij} \geq y_h, \quad \forall \mathcal{S} \subset \mathcal{V}, 2 \leq |\mathcal{S}| \leq |\mathcal{V}| - 2, h \in \mathcal{S}. \tag{1.56}$$

On the other side, Constraints (1.55) are still valid for non-HRP but may be strengthened by taking advantage of the variables y_i as follows (Feillet *et al.* (2005)). We have that

$$\sum_{i\in\mathcal{S}} y_i = \sum_{(i,j)\in\gamma(\mathcal{S})} x_{ij} + \sum_{(i,j)\in\delta^+(\mathcal{S})} x_{ij}. \tag{1.57}$$

Using Equation (1.56) we obtain:

$$\sum_{(i,j)\in\gamma(\mathcal{S})} x_{ij} \leq \sum_{v\in\mathcal{S}\setminus\{h\}} y_v, \quad \forall \mathcal{S} \subset \mathcal{V}, 2 \leq |\mathcal{S}| \leq |\mathcal{V}| - 2, h \in \mathcal{S}. \tag{1.58}$$

These constraints are valid for all the non-HRP presented in this section and can be separated in polynomial time by solving a max-flow problem.

1.3.8 Single-vehicle non-HRP with time windows

A classical constraint in the routing problems imposes that the visit of $i \in \mathcal{V}$ takes place during a time interval, called *time window* (TW). As a consequence, a TW expressed as $[E_i, L_i]$ is associated with each location $i \in \mathcal{V}$. Arriving at the location before E_i is feasible, but impose to wait until E_i to start the service. On the other side, arriving after L_i is not allowed. The previous models may slightly be modified to take these constraints into account by adding

$$E_i y_i \leq u_i \leq L_i y_i, \quad \forall i \in \mathcal{V}. \tag{1.59}$$

and by replacing the subtour elimination constraints in the MTZ form by

$$u_i - u_j + T_{ij} x_{ij} \leq M(1 - x_{ij}), \qquad \forall i, j \in \mathcal{V}, j \neq 0, i \neq j, \tag{1.60}$$

$$u_i + T_{i0} x_{i0} \leq L_0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.61}$$

where M is a large enough value. Here the variables u_i determine the time at which the service at location $i \in \mathcal{V}$ starts rather than the position on the tour.

1.4 Multi-vehicle case

This second part of the paper focuses on non-HRP where a fleet $\mathcal{F}$ of F vehicles (a group of people, multiple rings, etc.) is in charge of the servicing operation, $\mathcal{F} = \{1, 2, ..., F\}$. A capacity Q is associated with each of the vehicles that are supposed to be identical. As in the previous section, we begin this section by introducing the Capacitated Vehicle Routing Problem (CVRP). Then we present different families of non-HRP that derive from the CVRP in which the Hamiltonian requirement is removed. These problems include the Generalized Vehicle Routing Problem, the Generalized Vehicle Routing Problem with Time Windows, VRP with Profits, the Multi-vehicle CTP, the Capacitated Multiple RSP, and the Multi-vehicle TPP.

1.4.1 The Capacitated Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one family of problems well-studied in operations research. The most classical version is the Capacitated VRP (CVRP) which consists in finding at most F routes to visit all customer locations such that routing

costs are minimized and capacity constraints are respected. Each customer $i \in \mathcal{V} \setminus \{0\}$ has a demand Q_i.

It can be formulated using variables x_{ij} and u_i introduced in Section 1.2.2 as follows:

$$(VRP) \quad \min \sum_{(i,j)\in\mathcal{A}} C_{ij}x_{ij} \tag{1.62}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = 1 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{1.63}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = 1 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{1.64}$$

$$u_i - u_j + Qx_{ij} \leq Q - Q_j \qquad \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.65}$$

$$\sum_{(0,j)\in\delta^+(0)} x_{0j} \leq F, \tag{1.66}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j, \tag{1.67}$$

$$Q_i \leq u_i \leq Q \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.68}$$

The objective function (1.62) minimizes the total routing cost of the at most F routes. Constraints (1.63) and Constraints (1.64) are flow conservation constraints and impose that each vertex is visited exactly once. Constraints (1.65) ensure that the capacities of the vehicles are respected and are subtour elimination constraints in the MTZ form. Constraints (1.66) impose that at most F arcs leave the depot, namely at most F routes are allowed to accomplish service. Constraints (1.67) and Constraints (1.68) define the variables. In particular, Constraints (1.68) are capacity constraints as well. The interested reader is refereed to Golden *et al.* (2008); Laporte (2009); Toth & Vigo (2014).

1.4.2 The Generalized Vehicle Routing Problem

The multi-vehicle case of GTSP is the Generalized Vehicle Routing Problem (GVRP) and was introduced by Ghiani & Improta (2000). In the GVRP, the vertices of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ are partitioned into clusters, i.e., $\mathcal{C}_0 = \{0\}, \mathcal{C}_1, ..., \mathcal{C}_K$ clusters. $\mathcal{C}_0 \bigcup ... \bigcup \mathcal{C}_K = \mathcal{V}$ and $\mathcal{C}_h \cap \mathcal{C}_k = \emptyset, \forall h, k \in \mathcal{K}, h \neq k$, where $\mathcal{K} = \{0, 1, ..., K\}$ denotes the cluster index set. Cluster $\mathcal{C}_0 = \{0\}$ contains only the depot 0 where the fleet of vehicles is located. Each vehicle has a capacity of Q to perform deliveries. Each cluster is associated with a demand Q_k. At the depot the demand is $Q_0 = 0$.

The arc set $\mathcal{A}$ contains arcs that link vertices belonging to different clusters, that is, $\mathcal{A} = \{(i,j) | i \in \mathcal{C}_k, j \in \mathcal{C}_l, k \neq l, k, l \in \mathcal{K}\}$.

The GVRP consists of finding a set of at most F vehicle routes on $\mathcal{G}$ such that the traveling cost is minimized and: (i) every route starts and ends at the depot; (ii) exactly one vertex from each cluster is visited by a single vehicle; (iii) the sum of the demands of customers served by the same vehicle does not exceed Q.

Before presenting the mixed integer linear programming formulation for the GVRP, let us introduce the following notation. $Id(i)$ denotes the index of the cluster that contains vertex i, thus $i \in \mathcal{C}_k \Leftrightarrow Id(i) = k$.

The GVRP can be formulated using variables x_{ij}, y_i and u_i introduced in Section 1.2.2 as follows:

$$(GVRP) \quad \min \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij} \tag{1.69}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{1.70}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \forall i \in \mathcal{V}, \tag{1.71}$$

$$\sum_{i\in\mathcal{C}_k} y_i = 1 \qquad \forall k \in \mathcal{K}, \tag{1.72}$$

$$u_i - u_j + Q x_{ij} \leq Q - Q_{Id(j)} \qquad \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.73}$$

$$\sum_{j\in\mathcal{V}\setminus\{0\}} x_{0j} \leq F, \tag{1.74}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{V}, \tag{1.75}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, \tag{1.76}$$

$$Q_{Id(i)} \leq u_i \leq Q \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.77}$$

The objective function (1.69) minimizes the total routing cost. Constraints (1.70) and Constraints (1.71) are flow conservation constraints. Constraints (1.72) impose that exactly one vertex is visited per cluster. Constraints (1.73) ensure that the capacities of the vehicles are respected and are subtour elimination constraints in the MTZ form. Constraints (1.74) impose that at most F arcs leave the depot, namely at most F routes are allowed to accomplish service. Constraints (1.75)– (1.77) define the variables. Moreover, Constraints (1.77) are capacity constraints.

If all clusters $\mathcal{C}_i$, $i = 1, \ldots, K$ are singletons, the GVRP reduces to the VRP. From this observation it immediately follows, by reduction to the VRP, that the GVRP is NP-hard.

In the following, we summarize all the works that have been done for the GVRP.

Transformation to capacitated arc routing problem

Ghiani & Improta (2000) are the first to study the GVRP and propose a transformation of the GVRP into the capacitated arc routing problem (CARP), which allows the algorithms available for the latter to be used to solve the former. All the vertices in the same cluster are connected by a loop with edges having very large costs M and these edges are required edges in the corresponding CARP. If a cluster only has one vertex i inside, then a required edge (i, i) is introduced. The cost of each inter-cluster edge is increased by $M/2$ if an endpoint coincides with the depot, otherwise by M. By solving a CARP on the transformed graph, a solution for the GVRP can be obtained.

Exact methods

Kara & Bektas (2003) propose a compact integer linear programming formulation for the GVRP, adapting the well-known MTZ subtour elimination constraints for the TSP to the GVRP. Pop *et al.* (2012) provide two new compact formulations for the GVRP.

Bektaş *et al.* (2011) propose four integer linear programming formulations for the GVRP and develop an efficient branch-and-cut algorithm to solve it. They also apply an adaptive large neighborhood search heuristic to determine the upper bounds. A new data set for the GVRP containing 158 instances is generated, and is used as the benchmark for the GVRP hereafter. The results show that their branch-and-cut algorithm based on the best of the four formulations can solve instances with up to 121 nodes and 51 clusters. Reihaneh & Ghoniem (2018) developed a branch-cut-and-price algorithm for the GVRP. Their computational study indicated that the proposed algorithm is competitive with respect to the branch-and-cut algorithm proposed by Bektaş *et al.* (2011). Moreover it solved eight benchmark instances to optimality that were previously unsolved.

Ha *et al.* (2014) and Afsar *et al.* (2014) study a variant of the GVRP where the size of the fleet is flexible. Ha *et al.* (2014) propose a two-commodity flow formulation and a branch-and-cut method for the GVRP. Based on the results obtained on benchmark

instances, it can be stated that the branch-and-cut proposed by Ha *et al.* (2014) is more effective than the method described by Bektaş *et al.* (2011). Afsar *et al.* (2014) develop an exact method based on column generation for the GVRP with flexible fleet size.

Biesinger *et al.* (2016) study a variant of the GVRP, in which customers have stochastic demands. They propose an integer L-shaped method based on decomposition and branch-and-cut. Results show that this method is efficient in solving small instances up to about 40 vertices and 13 clusters.

Heuristic methods

Bautista *et al.* (2008) address a special case of the GVRP derived from an urban waste collection problem, in which each cluster contains at most two vertices. The authors propose two heuristics based on ant colonies and the results of the practical instances using the proposed heuristics obtained significant improvements. Pop *et al.* (2011) present constructive heuristics and local search algorithms for solving the GVRP, but without any computational experiments. Pop *et al.* (2013) present a hybrid algorithm combining a genetic algorithm and a local search procedure, and results show that it was competitive with the adaptive large neighborhood search proposed by Bektaş *et al.* (2011). Ha *et al.* (2014) propose a hybrid metaheuristic combining a greedy randomized adaptive search procedure with an evolutionary local search. Afsar *et al.* (2014) propose two metaheuristics based on a route-first cluster-second approach, in which the split procedure is executed using an iterated local search. Computational results show that their metaheuristics are very efficient, finding solutions with small optimality gap in a few seconds. The largest instances tackled by all these methods contains 262 vertices and 131 clusters.

There are several works studying the variants of the GVRP. For the GVRP with stochastic demands, Biesinger *et al.* (2015) propose a variable neighborhood search approach. It can identify optimal or near-optimal solutions for small instances in much shorter time than the exact method proposed by Biesinger *et al.* (2016), while it obtains large optimality gaps for medium and large instances. Biesinger *et al.* (2018) present a genetic algorithm combined with a variable neighborhood search. According to the computational results, it is superior to the algorithm described by Biesinger *et al.* (2015).

Zhou *et al.* (2018) introduce a city logistics problem called the multi-depot two-echelon VRP with delivery options. In the second level of the distribution network, customers are provided with different delivery options, allowing them to retrieve their packages at pick-up points. Thus, the second level can be formulated as a GVRP, in which the different delivery options of a customer forms a cluster.

Moccia *et al.* (2012) study what they called the Generalized VRPTW. They define a TW for each cluster. The authors present an incremental tabu search heuristic for the problem and assess the efficiency of the method by testing it on the GVRP instances and multi-depot VRPTW instances.

Applications

The GVRP has many applications, such as urban waste collection problem (Bautista *et al.*, 2008), the vessels routing in maritime transportation, healthcare logistics, the survivable telecommunication network design, etc. (Bektaş *et al.*, 2011). For example, in the routing of vessels in maritime transportation, a number of regions is given, each with several ports where the cargo can be delivered. If ships only need to deliver the cargo to one single port in each region, then the corresponding routing problem can be modeled as a GVRP, where the regions correspond to the clusters and the fleet of vessels corresponds to the fleet of vehicle (Bektaş *et al.*, 2011). Baldacci *et al.* (2010) also mention that problems like the TSP with profits, the VRP with selective backhauls, the covering VRP, the windy routing problem, etc., can be modeled as GVRPs.

1.4.3 The GVRP with Time Windows

The GVRP can be extended to the GVRP with Time Windows (GVRPTW) if we associate a time window (TW) $[E_i, L_i]$ with each vertex $\forall i \in \mathcal{V}$. The TW associated with the depot, i.e., $[E_0, L_0] = [0, T]$ represents the overall time horizon. A visit can only be made to a vertex during its TW, and an early arrival leads to a waiting time while a late arrival causes infeasibility. The objective of the GVRPTW consists of finding a set of at most F vehicle routes on $\mathcal{G}$ such that the traveling cost is minimized and: (i) every route starts and ends at the depot during $[0, T]$; (ii) exactly one vertex from each cluster is visited by a single vehicle; (iii) the sum of the demands of customers served by the same vehicle does not exceed Q; (iv) the service at vertex i starts during its TW $[E_i, L_i]$.

When the TWs associated with the locations of the same cluster do not overlap the problem is called VRP with roaming delivery locations (VRPRDL). The VRPRDL was introduced by Reyes *et al.* (2017). It is inspired by the trunk/in-car delivery, i.e., customers' packages can be delivered to the trunks of their cars.

Besides variables x_{ij}, y_i and u_i used for the GVRP, variables $t_i \in \mathbb{R}_+$ are introduced to represent the service time at vertex $i \in \mathcal{V}$. Then, the formulation for the GVRPTW can be obtained by adding the following constraints to the formulation of the GVRP defined by (1.69) – (1.77).

$$t_i - t_j + T_{ij}x_{ij} \leq M(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A}, j \neq 0, \tag{1.78}$$

$$E_i y_i \leq t_i \leq L_i y_i \qquad \forall i \in \mathcal{V}, \tag{1.79}$$

$$t_i + T_{i0}x_{i0} \leq L_0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.80}$$

Constraints (1.78) and (1.79) determine the starting time of service at each vertex and ensure that the TW are respected on the visited location. Constraints (1.79) also eliminate subtours since they generalize the subtour elimination constraints of Miller, Tucker and Zemlin for the TSP (Miller *et al.*, 1960). Constraints (1.80) ensure that all the vehicles return to the depot before the end of its TW.

The GVRPTW is NP-hard since it reduces to the GVRP when all the TW are set to $[0, +\infty]$. In the following, we summarize all the works that have been done for the VRPRDL.

Exact methods

Ozbaygin *et al.* (2017) develop a branch-and-price algorithm for the VRPRDL. Computational results on benchmark instances show that the proposed algorithm is able to solve to optimality instances with up to 60 clusters in a few minutes. For most of the large instances with 120 clusters, the algorithm is not able to prove optimality on a 6 hours time computation budget. Moreover, they also provide another set of instances for a hybrid delivery strategy combining trunk delivery and home delivery. In these instances, in each cluster the TW associated with the home location corresponds to the planning horizon and overlaps all other TWs, while the other TW associated with the trunk locations are non-overlapping. This instance set also has a specific TW structure. The results revealed that using this combined strategy led to an average cost savings

of nearly 20% compared to the classical delivery system when only home delivery is available.

Following this work, Ozbaygin & Savelsbergh (2018) introduce a dynamic variant of the VRPRDL, in which customer itineraries may change during the execution of a planned delivery schedule. The branch-and-price algorithm proposed in Ozbaygin *et al.* (2017) is used to obtain the planned delivery schedule based on initial customer itineraries, as well as the reoptimization solutions whenever a customer itinerary change is revealed. To ensure computational efficiency when solving reoptimization problems, they reuse and suitably modify the columns generated during previous branch-and-price runs.

Heuristic methods

The work of Reyes *et al.* (2017) is the first to study the VRPRDL and proposes a construction heuristic based on a greedy randomized adaptive search procedure, and an improvement heuristic. The results show the economic advantages for the delivery companies to consider trunk deliveries instead of the traditional home delivery.

Lombard *et al.* (2018) study a variant of the VRPRDL with stochastic travel times. Instead of using deterministic travel times, the authors use a matrix of probability distribution, which indicates the distribution of travel times between two locations. They use a combination of a Monte-Carlo method and a greedy randomized adaptive search procedure. This approach can obtain delivery solutions for small-sized instances.

Applications

The GVRPTW models the situation where a customer is associated with several delivery locations. He/she specifies the time intervals at which he/she is available to receive the parcel. The VRPRDL arises in the last mile delivery with trunk/in-car delivery option (Ozbaygin *et al.*, 2017; Reyes *et al.*, 2017). Volvo launched its world-first in-car delivery service in Sweden in 2016 (Kirsten, 2016). It is a service for delivering goods directly to one customer's Volvo car. This is achieved by electronically delegating one-time access to an authorized delivery company to the customer's car. This enables the customer to select his/her Volvo as the delivery location and to track the delivery when it happens. In April 2018, Amazon also launched the in-car service in partnership with

two major automakers General Motors and Volvo. This service is available in 37 cities across the US (Hawkins, 2018).

1.4.4 The VRP with Profits

The multi-vehicle case of TSPPs is the VRP with Profits (VRPPs), where in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ each vertex $i \in \mathcal{V} \setminus \{0\}$ is associated with a profit P_i. The objective is to optimize the collected profit and the traveling cost, thus not all the vertices need to be visited. The goal of the VRPPs is to find a set of routes starting and ending at a depot which visit a subset of vertices such that an objective function is optimized. As in the TSPPs, the objective function of the VRPPs can be expressed in different ways given hereafter. To be consistent with the definitions of the TSPPs, in this survey we straightforwardly extend the classification of the TSPPs to VRPPs as follows:

- The Profitable VRP where the objective is to find a set of tours that minimize the traveling cost minus the collected profit.
- The Team Orienteering Problem (TOP, Chao *et al.* (1996b)) where the objective is to find a set of tours that maximizes the profit under a constraint that imposes the maximum duration T_{max} of a tour.
- The Prize-Collecting Vehicle Routing Problem (PCVRP) where the objective is to find a set of tours that minimizes the traveling cost under a constraint that imposes a minimum prize collection P_{min} of a tour.

Let x_{ijf} be a binary variable that equals 1 if and only if arc (i, j) is traversed by vehicle f in the solution. Let y_{if} be a binary variable that equals 1 if and only if vertex $i \in \mathcal{V}$ is visited by vehicle f in the solution.

The VRPPs can be formulated as follows:

$$(VRPPs) \quad \min f(x, y) \tag{1.81}$$

$$\text{s.t.} \sum_{f \in \mathcal{F}} \sum_{(i,j) \in \delta^+(i)} x_{ijf} = \sum_{f \in \mathcal{F}} y_{if} \qquad \forall i \in \mathcal{V}, \tag{1.82}$$

$$\sum_{f \in \mathcal{F}} \sum_{(j,i) \in \delta^-(i)} x_{jif} = \sum_{f \in \mathcal{F}} y_{if} \qquad \forall i \in \mathcal{V}, \tag{1.83}$$

$$\sum_{f \in \mathcal{F}} y_{if} \leq 1 \qquad i \in \mathcal{V} \setminus \{0\}, \tag{1.84}$$

$$u_i - u_j + N \sum_{f \in \mathcal{F}} x_{ijf} \leq N - 1 \qquad \forall i, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.85}$$

$$\sum_{f \in \mathcal{F}} \sum_{j \in \mathcal{V} \setminus \{0\}} x_{0jf} \leq F, \tag{1.86}$$

$$g(x, y) \geq 0 \qquad \forall f \in \mathcal{F}, \tag{1.87}$$

$$x_{ijf} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j, f \in \mathcal{F}, \tag{1.88}$$

$$y_{if} \in \{0, 1\} \qquad \forall i \in \mathcal{V}, f \in \mathcal{F}, \tag{1.89}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.90}$$

where $f(x, y)$ and $g(x, y)$ are as follows.

- for the Profitable VRP:
  - $f(x, y) = \sum_{f \in \mathcal{F}} \sum_{(i,j) \in \mathcal{A}} C_{ij} x_{ijf} - \sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{V} \setminus \{0\}} P_i y_{if}$
  - $g(x, y) = 0$;
- for the TOP:
  - $f(x, y) = -\sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{V} \setminus \{0\}} P_i y_{if}$
  - $g(x, y) = T_{max} - \sum_{(i,j) \in \mathcal{A}} T_{ij} x_{ijf}$;
- for the PCVRP:
  - $f(x, y) = \sum_{f \in \mathcal{F}} \sum_{(i,j) \in \mathcal{A}} C_{ij} x_{ijf}$
  - $g(x, y) = \sum_{i \in \mathcal{V}} P_i y_i - P_{min}$.

The objective function (1.81) is to minimize the traveling cost minus the collected profit in the case of the profitable VRP, is to maximize the profit in the case of the TOP and is to minimize the traveling cost in the case of the PCVRP. Constraints (1.82) and (1.83) are flow conservation constraints and ensure that an arc enters and exits each selected vertex. Constraints (1.84) guarantee that each vertex is visited at most once. Constraints (1.85) are MTZ type of the subtour elimination constraints. Constraints (1.86) impose that the number of routes cannot exceed the number of available vehicles. Constraints (1.87) are inactive for the Profitable VRP, impose the maximum tour duration for the TOP and the minimum prize to collect for the PCVRP. Constraints (1.88) – (1.90) are variable definitions.

The VRPPs are NP-hard since the TSPPs are NP-hard.

Note that in the well-known TOP, there is a maximum duration constraint for each route. It was first introduced by Butt & Cavalier (1994) under the name multiple tour maximum collection problem while it is named as the TOP by Chao *et al.* (1996b). However, even though some variants of the profitable VRP and PCVRP have been studied, there is no fixed and commonly used definitions for both problems. Moreover, notice that in the literature, the use of names Profitable VRP and *PCVRP* can be mixed. Thus, in this paper, the works are classified according to the objective function of the studied problem as mentioned above.

The interested reader is referred to the following survey papers. The article of Archetti *et al.* (2014) is dedicated to the VRPPs, including research works on the single-vehicle as well as the multi-vehicle case, up to the year 2014. For the multi-vehicle case, the authors mainly focus on the TOP and its variants and on the VRP with private fleet and common carrier. They mention two works on the profitable VRP and no work on the PCVRP.

Vansteenwegen *et al.* (2011) provide a comprehensive survey on the OP and its variants, also covering works for the TOP and its variants up to the year 2009. Gunawan *et al.* (2016) review the literature on the OP including new works and new variants up to 2015. They also consider works on the TOP and its variants.

Compared with the TOP, much fewer studies can be found on the profitable VRP and PCVRP. Therefore, for these two problems, we summarize all the works that have been published in the literature. For the TOP, in order to avoid the repetition with the survey mentioned above, we focus on the related literature and its variants published after 2015.

Exact methods: Profitable VRP

All the works in the following study variants of the profitable VRP. Archetti *et al.* (2009) introduce the capacitated versions of the profitable VRP (which they call as the capacitated PTP) and the TOP where a fleet of capacitated vehicles is available. They propose exact methods based on column generation that can solve small size instances. They also propose two variants of tabu search algorithm and a variable neighborhood search algorithm, which can obtain very good results for both problems.Archetti *et al.* (2013) propose a branch-and-price algorithm for the capacitated profitable VRP and the capacitated TOP. A heuristic is embedded in the exact approach to find good

feasible solutions quickly. Computational results show that several unsolved benchmark instances are solved to optimality.

Archetti *et al.* (2017) introduce and study the undirected capacitated general routing problem with profits. It is defined on an undirected graph in which customer profit can be collected from some of the vertices as well as from some of the edges. The authors develop a branch-and-cut algorithm for this problem.

Orlis *et al.* (2019) study the capacitated VRP with profits and service level requirements arising in a cash supply chain in the Netherlands. The service level requirement of a customer is the minimum-accepted percentage of fulfilled requests over their total number. When the requirement is not met, a predefined penalty is applied. The authors propose a branch-and-cut algorithm by adapting several valid inequalities proposed in the literature.

Heuristic methods: Profitable VRP

All the works in the following study variants of the profitable VRP. Tang & Wang (2006) consider a problem arising in the hot rolling production in the steel industry, which is modeled as a capacitated profitable VRP with an additional constraint. Each vertex is associated with a demand and each vehicle with a capacity. The additional constraint is on the total demand of the visited vertices which must not be less than a predefined amount. The objective is a linear combination of three objectives, i.e., the minimization of total traveled distance, the minimization of the number of vehicles used, and the maximization of the profit that is collected. An iterated local search algorithm based on a large-scale neighborhood is proposed. Li & Tian (2016) study the same problem but with a different objective, which consists in the minimization of the transportation cost minus the profit. A self-adaptive variable neighborhood search algorithm is proposed. Zhang *et al.* (2009) consider the same problem but consider each objective separately. Thus they study the multi-objective version. A particle swarm optimization algorithm is developed.

Aras *et al.* (2011) study a problem where a firm collects cores from its dealers, and each visit to a dealer is associated with a gross profit and an acquisition price to be paid to take the cores back. This is a multi-depot profitable VRP. The objective is the maximization of the revenue from the cores minus the total cost of purchasing cores and operating the vehicles. Two mixed integer linear programming formulations are proposed and a tabu search based heuristic is developed.

Chbichib *et al.* (2012) consider a profitable VRP with multiple trips. Each vehicle is allowed to perform several routes under a workday duration limit. They propose four formulations for this problem that are compared using CPLEX on small-size instances. They also propose construction and improvement heuristics in order to solve large-size instances.

Lahyani *et al.* (2013) address a problem where customer requests include several products and multi-compartment vehicles are used. A profit is associated with each product and the customer can be delivered with only part of his request. It is a variant of the capacitated profitable VRP with time windows and incompatibility constraints. The authors propose a variable neighborhood search algorithm for this problem.

Vidal *et al.* (2015) propose new large neighborhoods for the profitable VRP as well as for TOP and these neighborhoods contribute to finding solutions of higher quality compared with the previous state-of-the-art methods. Fifty-two new best-known solutions have been found.

Gansterer *et al.* (2017) study the multi-vehicle profitable pickup and delivery problem, where multiple carriers transport goods from a selection of pickup customers to the corresponding delivery customers within given travel time limits. Two variable neighborhood search heuristics are developed.

Stavropoulou *et al.* (2019) study the profitable VRP with consistency constraints. It takes into account the fact that customers service should be provided in a consistent manner in order to increase brand loyalty and customer satisfaction. The authors propose an adaptive tabu search algorithm to solve this problem.

Exact methods: TOP

El-Hajj *et al.* (2016) present a cutting plane algorithm to solve the TOP. Several types of cuts are proposed. Computational results show that the proposed approach is competitive and is able to prove the optimality for 12 instances previously unsolved.

Gedik *et al.* (2017) use constraint programming to formulate and solve the TOP with time windows (TOPTW) by applying interval variables. This approach identifies one new best-known solution for TOPTW benchmark instances and solves two more instances to optimality.

Heuristic methods: TOP

Tsakirakis *et al.* (2019) propose a harmony search for the TOP which is inspired from the composition of music harmonies. The proposed algorithm with dynamic adjustment of the parameters is superior to the static version using predefined values of parameters. Results show that the proposed algorithms are competitive with the other efficient algorithms described in the literature.

Ben-Said *et al.* (2019) study the capacitated TOP. Their algorithm alternates between two search spaces, i.e., the giant tour and routes search spaces, under the framework of a hybrid heuristic combining greedy randomized adaptive search procedure and evolutionary local search. Computational results show the efficiency of the algorithm.

There are several works dealing with the TOPTW and its variants. Lin & Vincent (2017) study the TOPTW with mandatory visits: some customers considered as important must be visited. The authors propose a multi-start simulated annealing heuristic to solve it. Vincent *et al.* (2017) address the multi-modal TOPTW which is motivated by a tourist trip design application where multiple transportation modes are available for tourists. A two-level particle swarm optimization algorithm with two solution representations and decoding methods are proposed.

Hu *et al.* (2018) study a multi-objective TOPTW in which multiple profits are associated with one node. A multi-objective evolutionary algorithm based on decomposition and constraint programming is proposed. Computational results show that many new non-dominated solutions are found. Hapsari *et al.* (2019) study a multi-objective TOPTW in which one objective is to maximize the profit and the other is to minimize the time needed for the tourist's itinerary. The authors propose a metaheuristic based on iterated local search.

Gavalas *et al.* (2019) point out the weakness of the state-of-the-art metaheuristic for the TOPTW, i.e., the iterated local search (Vansteenwegen *et al.*, 2009). The authors propose two cluster-based extensions to ILS by grouping nodes on separate clusters based on geographical criteria. Computational results show that the proposed algorithms outperform ILS in terms of solution quality and computation time.

Vincent *et al.* (2019) study the TOPTW with time-dependent scores where the score of visiting a node is different depending on the time of visit. A hybrid artificial bee colony algorithm is proposed, which embeds the acceptance criterion of simulated annealing.

Heuristic methods: PCVRP

Stenger *et al.* (2013) consider the PCVRP with non-linear cost in its one and multi-depot variants, which integrates the option of outsourcing customers to subcontractors instead of serving them with the private fleet. The objective is to minimize the total cost. An adaptive variable neighborhood search algorithm is proposed.

Applications

VRPPs have been well studied and model a variety of applications. One application of VRPPs arises in the context of the small packaging shipping (SPS) industry. Large companies outsource last-mile deliveries of unprofitable areas to subcontractors and pay subcontractors per parcel delivered, which is independent from routing decisions of subcontractors. Therefore, for SPS companies, not all parcels must be delivered by themselves. They can select a subset of parcels to deliver to minimize the overall cost. It is called the VRP with private fleet and common carrier (Archetti *et al.*, 2014).

Another well-known application is the tourist trip design problem which aims to maximize tourist satisfaction (profit) of the visited attractions in a limited period while satisfying some practical constraints. The interested reader is referred to the survey on the tourist trip design problem by Gavalas *et al.* (2014).

Some problems arising in the hot rolling production of the steel industry can also be modeled as VRPPs (Li & Tian, 2016; Tang & Wang, 2006).

1.4.5 The Multi-vehicle Covering Tour Problem

The multi-vehicle case of the CTP (mCTP) is first introduced by Hachicha *et al.* (2000) considering the routing of mobile health care delivery teams in developing countries. As in the CTP, it is defined on a graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{W}, \mathcal{A})$, where $\mathcal{V} \cup \mathcal{W}$ is the vertex set, $\mathcal{V}$ is a set of vertices that can be visited, $\mathcal{T} \subseteq \mathcal{V}$ is a set of vertices that must be visited and $\mathcal{W}$ is a set of vertices that must be covered, i.e., that must lie within a prespecified distance D from a visited vertex on the tour. Vertex $0 \in \mathcal{T}$ is the depot at which are based a fleet $\mathcal{F}$ of F homogeneous vehicles, $\mathcal{F} = \{1, 2, ..., F\}$.

The mCTP consists of determining a set of at most F routes of minimum total length over a subset of $\mathcal{V}$ such that (i) every route starts and ends at the depot; (ii) each vertex in $\mathcal{T}$ belongs to exactly one route, while each vertex in $\mathcal{W}$ must be covered by a vertex visited on one of the routes; (iii) the number of vertices on any route

(excluding the depot) cannot exceed a preset value P_r, and the length of any route cannot exceed a preset value L_r. They are referred as capacity constraints. Compared with the CTP, for each route, the mCTP have upper bounds for its length and the number of vertices visited respectively.

For arc $(i,j) \in \mathcal{A}$ and vehicle $f \in F$, let x_{ijf} be a binary variable equal to 1 if and only if arc (i,j) is traversed by vehicle f in the solution. For $i \in \mathcal{V}$ and $f \in \mathcal{F}$, let y_{if} be a binary variable equal to 1 if and only if vertex $i \in \mathcal{V}$ is visited by vehicle f in the solution. $\mathcal{S}_j = \{i \in \mathcal{V} \setminus \{0\} | D_{ij} \leq D\} \subset \mathcal{V}$ is the set of vertices that cover vertex $j \in \mathcal{W}$.

The mCTP can be formulated as follows:

$$(mCTP) \quad \min \sum_{f\in\mathcal{F}} \sum_{(i,j)\in\mathcal{A}} D_{ij}x_{ijf} \tag{1.91}$$

$$\text{s.t.} \sum_{f\in\mathcal{F}} \sum_{(i,j)\in\delta^+(i)} x_{ijf} = \sum_{f\in\mathcal{F}} y_{if} \qquad \forall i \in \mathcal{V}, \tag{1.92}$$

$$\sum_{f\in\mathcal{F}} \sum_{(j,i)\in\delta^-(i)} x_{jif} = \sum_{f\in\mathcal{F}} y_{if} \qquad \forall i \in \mathcal{V}, \tag{1.93}$$

$$\sum_{f\in\mathcal{F}} \sum_{i\in\mathcal{S}_j} y_{if} \geq 1 \qquad \forall j \in \mathcal{W}, \tag{1.94}$$

$$\sum_{f\in\mathcal{F}} y_{if} = 1 \qquad \forall i \in \mathcal{T}, \tag{1.95}$$

$$\sum_{f\in\mathcal{F}} y_{if} \leq 1 \qquad i \in \mathcal{V} \setminus \{0\}, \tag{1.96}$$

$$u_i - u_j + N \sum_{f\in\mathcal{F}} x_{ijf} \leq N - 1 \qquad \forall i,j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.97}$$

$$\sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{V}\setminus\{0\}} x_{0jf} \leq F, \tag{1.98}$$

$$\sum_{i\in\mathcal{V}\setminus\{0\}} y_{if} \leq P_r \qquad \forall f \in \mathcal{F}, \tag{1.99}$$

$$\sum_{(i,j)\in\mathcal{A}} D_{ij}x_{ijf} \leq L_r \qquad \forall f \in \mathcal{F}, \tag{1.100}$$

$$x_{ijf} \in \{0,1\} \qquad \forall i,j \in \mathcal{V}, i \neq j, f \in \mathcal{F}, \tag{1.101}$$

$$y_{if} \in \{0,1\} \qquad \forall i \in \mathcal{V}, f \in \mathcal{F}, \tag{1.102}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \tag{1.103}$$

The objective function (1.91) is to minimize the total traveled length. Constraints (1.92)

and (1.93) are flow conservation constraints and impose that an arc enters and exits each selected vertex. Constraints (1.94) make sure that every vertex in $\mathcal{W}$ is covered by the routes. Constraints (1.95) guarantee that every vertex in $\mathcal{T}$ is visited by the routes. Constraints (1.96) guarantee that each vertex in $\mathcal{V}$ belongs to at most one route. Constraints (1.97) are subtour elimination constraints. Constraints (1.98) ensure that at most F vehicles enter and leave the depot. Constraints (1.99) impose that the number of vertices visited in each route should not exceed P_r. Constraints (1.100) impose that the length of each route should not exceed L_r. Constraints (1.101) – (1.103) define the variables.

The mCTP is NP-hard since it reduces to a VRP with unit demands when $\mathcal{T} = \mathcal{V}$ and $\mathcal{W} = \emptyset$, or to a CTP when the capacity constraints are relaxed (Ha *et al.*, 2013). In the following, we summarize all the articles that have been published on the mCTP and its variants. To the best of our knowledge, there is no work dedicated to the multi-vehicle case of the CSP.

Exact methods

The first exact method for the mCTP is proposed by Lopes *et al.* (2013). They develop a branch-and-price algorithm, and specific dominance and pruning rules are introduced to accelerate the resolution of pricing problems. A column generation based heuristic is also described to determine upper bounds. Jozefowiez (2014) present a branch-and-price algorithm for the mCTP and provided computational results for randomly generated instances with up to $|\mathcal{V}| = 100$ and $|\mathcal{W}| = 150$. The Constraints (1.99) and (1.100) are considered in the subproblem, which is modeled as a RSP and solved by a branch-and-cut method.

Ha *et al.* (2013) study a variant of the mCTP where only the upper bounds on the number of vertices visited by a route are considered, and the constraints on the route length are relaxed, i.e., $L_r = +\infty$. The authors propose a two-commodity flow formulation and a branch-and-cut algorithm. A metaheuristic combining the greedy randomized adaptive search procedure and evolutionary local search is also developed.

Besides, Tricoire *et al.* (2012) study a variant of the mCTP, the bi-objective CTP with stochastic demands, in which demands are random variables with a known joint distribution. One of the objectives is to minimize the costs including routing costs for a fleet of vehicles and opening costs for distribution centers (the visited vertices).

The second objective is to minimize the uncovered demand. The authors proposed an epsilon-constraint algorithm involving branch-and-cut technique.

Karaoğlan *et al.* (2018) study the probabilistic mCTP. Its objective is to determine a set of distance-constrained routes maximizing the expected demand covered through visiting a subset of facilities, where a vehicle visiting a facility covers the demand of a customer with a probability in $[0, 1)$. A branch-and-cut algorithm is developed as well as a local search heuristic based on variable neighborhood search to obtain upper bounds.

Heuristic methods

Hachicha *et al.* (2000) introduce the mCTP, and propose an integer linear programming formulation and three heuristic algorithms. Kammoun *et al.* (2017) study the mCTP without the constraints on the route length and propose a variable neighborhood search heuristic, which outperforms the metaheuristic proposed by Ha *et al.* (2013).

Some works are devoted to variants of the mCTP. Naji-Azimi *et al.* (2012a) tackle the location of satellite distribution centers to provide humanitarian aid to the victims in a disaster area. They consider multiple commodities, heterogeneous capacitated fleet and split deliveries. A multi-start heuristic is proposed to solve the problem and it obtains high-quality solutions in reasonable computation times.

Oliveira *et al.* (2015) model the multi-vehicle urban patrolling problem as a mCTP, where there is no vehicle capacity constraint but a balance requirement among the vehicles. The authors propose several heuristics.

Allahyari *et al.* (2015) consider the multi-depot capacitated mCTP. The authors present two mixed integer programming formulations and a hybrid metaheuristic combining greedy randomized adaptive search procedure, iterated local search and simulated annealing.

Flores-Garza *et al.* (2017) introduce the cumulative mCTP, whose objective is to minimize the sum of arrival times (latency) at each visited location. There is a time limit on the duration of each tour. The authors propose a mixed integer linear programming formulation and a greedy randomized adaptive search procedure for the problem.

Pham *et al.* (2017) study a variant called the multi-vehicle multi-covering tour problem, in which a vertex must be covered several times rather than once. An integer linear programming formulation is presented for a special case of the problem, and a

branch-and-cut algorithm is proposed. The authors develop a genetic algorithm and computational results show that it outperforms the current best metaheuristics for several mCTP problems.

Murakami (2018b) study the mCTP in which a demand is assigned to each vertex, and each vehicle has a capacity. The sum of demands of any route should not exceed the vehicle capacity. The authors propose a column generation based heuristic.

Applications

One application of this problem arises in the VRP in Humanitarian Relief (Balcik *et al.*, 2008; Kovács & Spens, 2007; Luis *et al.*, 2012; Shaelaie *et al.*, 2014; Toth & Vigo, 2014), as goods and services are often delivered to central locations visited by beneficiaries. For example, in the disaster relief problem in Doerner & Hartl (2008), after a disaster the relief vehicles stop at several locations and the populations (the set $\mathcal{W}$) must visit one of the vehicle stops. The appropriate stops among $|\mathcal{V}|$ potential locations need to be chosen so that all populations can reach one of these stops within acceptable time. $\mathcal{T}$ can be considered as the set of stops covering the populations that cannot be covered by other stops. Another example is to supply the humanitarian aid to the affected people through several satellite distribution centers located within a predefined distance from their domiciles (Naji-Azimi *et al.*, 2012a). Another application arises in bi-level transportation networks, for example the postbox location problem (Labbé & Laporte, 1986). Last, in the routine patrol routing planning (Oliveira *et al.*, 2015), routes need to guarantee visibility which has an influence on the community safety, providing surveillance and allowing quick emergency responses. Vehicles available for the patrol are limited and strive to achieve balanced routes. This problem is modeled as a mCTP, in which a subset of locations must be visited, whereas the other locations should be close enough to the planned routes.

1.4.6 The Capacitated Multiple Ring Star Problem

The Capacitated Multiple RSP (CmRSP) was introduced by Baldacci *et al.* (2007). It is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The set of vertices $\mathcal{V} = \{0\} \cup \mathcal{V}'$, where 0 is the depot, $\mathcal{V}' = \mathcal{U} \cup \mathcal{W}$, $\mathcal{U}$ is the set of customers and $\mathcal{W}$ is the set of transition points (also called Steiner nodes). The arc set $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$. Each customer $i \in \mathcal{U}$ can be directly assigned (connected) to a subset of nodes $\mathcal{S}_i \subset \mathcal{U} \cup \mathcal{W}$. The arc

set $\mathcal{A}' = \{(i,j) : i \in \mathcal{U}, j \in \mathcal{S}_i\}, \mathcal{A}' \subset \mathcal{A}$ is the set of all possible connections and each arc in $\mathcal{A}'$ has a non-negative assignment/connection cost A_{ij}. Two input parameters F and Q are given, representing the number of rings and the capacity of each ring respectively.

A ring $\mathcal{R}$ corresponds to a route visiting a subset of vertices in $\mathcal{V}$ including the depot. There may be a number of customers $i \notin \mathcal{R}$ that are connected to vertices $j \in \mathcal{R}$ by arc $(i,j) \in \mathcal{A}'$. The objective of the CmRSP is to find a set of F rings starting and ending at the depot, such that each customer is assigned to exactly one ring, each Steiner node is visited at most once and the number of customers assigned to each ring does not exceed the ring capacity Q. The objective is to minimize the total routing and assignment costs.

For each arc $(i,j) \in \mathcal{A}$ and $f \in \mathcal{F}$, let x_{ijf} be a binary variable equal to 1 if and only if (i,j) belongs to a ring f in the solution. For each arc $(i,j) \in \mathcal{A}'$ and $f \in \mathcal{F}$, let y_{ijf} be a binary variable equal to 1 if and only if customer $i \in \mathcal{U}$ is assigned to vertex j on ring f. If a customer i is visited by a ring f, then it is assigned to itself, i.e., $y_{iif} = 1$.

The CmRSP can be formulated as follows:

$$(CmRSP) \quad \min \sum_{f\in\mathcal{F}} \sum_{(i,j)\in\mathcal{A}} C_{ij}x_{ijf} + \sum_{f\in\mathcal{F}} \sum_{\substack{(i,j)\in\mathcal{A}' \\ i\neq j}} A_{ij}y_{ijf} \tag{1.104}$$

$$\text{s.t.} \sum_{f\in\mathcal{F}} \sum_{(i,j)\in\delta^+(i)} x_{ijf} = \sum_{f\in\mathcal{F}} y_{iif} \qquad \forall i \in \mathcal{V}, \tag{1.105}$$

$$\sum_{f\in\mathcal{F}} \sum_{(j,i)\in\delta^-(i)} x_{jif} = \sum_{f\in\mathcal{F}} y_{iif} \qquad \forall i \in \mathcal{V}, \tag{1.106}$$

$$\sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{S}_i} y_{ijf} = 1 \qquad \forall i \in \mathcal{U}, \tag{1.107}$$

$$\sum_{f\in\mathcal{F}} y_{iif} \leq 1 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{1.108}$$

$$y_{ijf} \leq y_{jjf} \qquad \forall (i,j) \in \mathcal{A}', f \in \mathcal{F}, \tag{1.109}$$

$$\sum_{(i,j)\in\mathcal{A}'} y_{ijf} \leq Q \qquad \forall f \in \mathcal{F} \tag{1.110}$$

$$u_i - u_j + N \sum_{f\in\mathcal{F}} x_{ijf} \leq N - 1 \qquad \forall i,j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{1.111}$$

$$\sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{V}\setminus\{0\}} x_{0jf} = F, \tag{1.112}$$

$$x_{ijf} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, f \in \mathcal{F}, \quad (1.113)$$

$$y_{ijf} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}', f \in \mathcal{F}, \quad (1.114)$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V} \setminus \{0\}. \quad (1.115)$$

The objective function (1.104) is to minimize routing and assignment costs. Constraints (1.105) and (1.106) are flow conservation constraints. Constraints (1.107) guarantee that a customer is either on a ring or is assigned to a vertex on a ring. Constraints (1.108) impose that each vertex is visited by at most one ring. Constraints (1.109) impose that a customer can only be assigned to a vertex visited on the ring. Constraints (1.110) ensure that the number of customers assigned to each ring does not exceed the ring capacity Q. Constraints (1.111) are subtour elimination constraints. Constraints (1.112) ensure that F rings enter and leave the depot. Constraints (1.113) – (1.115) define the variables.

The CmRSP is NP-hard because when $\mathcal{W} = \emptyset$, $Q = |\mathcal{V}|$, $F = 1$ and the assignment costs are very high compared to the routing costs, it reduces to the TSP (Baldacci *et al.*, 2007). In the following, we summarize all the papers devoted to the CmRSP and its variants.

Exact methods

Baldacci *et al.* (2007) introduce the CmRSP and present two integer programming formulations for it. Valid inequalities are proposed and used as cutting planes in a branch-and-cut method. Hoshino & de Souza (2008) propose a set covering model for the CmRSP and develop a branch-and-price algorithm, which is competitive with the branch-and-cut approach proposed by Baldacci *et al.* (2007). Then Hoshino & De Souza (2012) extend their branch-and-price algorithm to a branch-cut-and-price algorithm by adding a subset of cuts proposed by Baldacci *et al.* (2007). The proposed algorithm provides a better bound at the root node than the branch-and-cut method, and outperforms the latter in several classes of instances. Some instances with up to 102 nodes are solved to optimality by both branch-and-cut (Baldacci *et al.*, 2007) and branch-cut-and-price (Hoshino & De Souza, 2012). Baldacci *et al.* (2017) propose pricing strategies based on dynamic programming algorithms for the CmRSP. Five different pricing strategies based on three different ring-star relaxations are presented. Computational results show that tight lower bounds can be computed for the CmRSP

instances with up to 431 nodes and for multi-depot RSP instances with up to 203 nodes and 3 depots.

Sundar & Rathinam (2017) study the multi-depot RSP. The objective is to find a set of routes (rings) minimizing the routing costs and assignment costs, such that each route passes through a set of vertices and exactly one depot, meanwhile each non-visited vertex is assigned to a visited vertex or a depot. A mixed integer linear programming formulation and some valid inequalities are proposed. The authors present a polyhedral analysis and derive facet-defining inequalities for the multi-depot RSP. A branch-and-cut algorithm is developed and evaluated on several classes of benchmark instances, with the largest solved instance involving 101 vertices.

Heuristic methods

Mauttone *et al.* (2007) propose a metaheuristic for the CmRSP, which is a hybrid algorithm combining a greedy randomized adaptive search procedure and a tabu search method. Naji-Azimi *et al.* (2010) propose a heuristic including a construction procedure and an improvement procedure. A series of different local search operations are applied iteratively in the improvement procedure. Naji-Azimi *et al.* (2012b) propose a variable neighborhood search that incorporates an integer linear programming based improvement method whenever the local searches are not able to improve the quality of the current solution. Zhang *et al.* (2014) propose a memetic algorithm for the CmRSP, which does not require the underlying graph satisfying the triangle inequality as in all previous works. The proposed approach obtains almost all best-known solutions for several benchmark instances with up to 431 nodes, making it the state-of-the-art heuristic for the CmRSP.

Several works address the multi-depot RSP through heuristics. Baldacci & Dell'Amico (2010) propose two construction heuristics, and use a tabu search algorithm to improve the solutions. Hill & Voß (2016) present a mataheuristic that iteratively refines a solution. In this approach, the subproblems are modeled as smaller instances of the global problem MDRSP, and a branch-and-cut method is used to solve them to optimality. Computational results show that 90% of the existing results are improved by the proposed approach for instances with up to 1000 nodes.

Applications

The CmRSP arises in the design of an urban optical telecommunication network for the city of Reggio Emilia, Italy (Baldacci *et al.*, 2007). In many fiber optical communication networks, in order to ensure the reliability, the ring topology is used since it prevents the loss of connection due to a single edge or a single-node failure. On the other hand, to reduce the excavation costs, if a customer is near to a ring, then it is allowed to connect this customer to the ring by a single edge. The new network topology is called ring-star structure. Xu *et al.* (1999) address a particular digital data service network design problem, which aims to interconnect a set of customer locations through a ring of end offices so as to minimize the total tariff cost and provide reliability (the RSP application).

1.4.7 The Multi-vehicle Traveling Purchaser Problem

The multi-vehicle TPP (mTPP) is first introduced by Choi & Lee (2010). It is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ and $\mathcal{V}\backslash\{0\}$ represents a set of markets. A fleet $\mathcal{F}$ of homogeneous vehicles with a limited capacity Q is available at the depot 0 for a set of purchasers collaborating to satisfy the products demand. The problem aims to minimize the purchasing and traveling costs deciding the purchasing plan and the corresponding visiting route for each vehicle.

For arc $(i, j) \in \mathcal{A}$ and vehicle $f \in F$, let x_{ijf} be a binary variable equal to 1 if and only if vehicle f visit vertex j immediately after i. For $i \in \mathcal{V}$ and $f \in \mathcal{F}$, let y_{if} be a binary variable equal to 1 if and only if vehicle f visits vertex i. For $i \in \mathcal{V} \setminus \{0\}$, $f \in \mathcal{F}$ and $q \in \mathcal{Q}$, let z_{ifq} be the quantity of product q purchased by vehicle f at vertex i.

The mTPP can be formulated as follows:

$$(mTPP) \quad \min \sum_{f\in\mathcal{F}} \sum_{(i,j)\in\mathcal{A}} C_{ij}x_{ijf} + \sum_{f\in\mathcal{F}} \sum_{i\in\mathcal{V}\setminus\{0\}} \sum_{q\in\mathcal{Q}} P_{iq}z_{ifq} \tag{1.116}$$

$$\text{s.t.} \sum_{f\in\mathcal{F}} \sum_{(i,j)\in\delta^+(i)} x_{ijf} = \sum_{f\in\mathcal{F}} y_{if} \qquad \forall i \in \mathcal{V}, \tag{1.117}$$

$$\sum_{f\in\mathcal{F}} \sum_{(j,i)\in\delta^-(i)} x_{jif} = \sum_{f\in\mathcal{F}} y_{if} \qquad \forall i \in \mathcal{V}, \tag{1.118}$$

$$\sum_{f\in\mathcal{F}} y_{if} \leq 1 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{1.119}$$

$$\sum_{f\in\mathcal{F}}\sum_{i\in V\setminus\{0\}} z_{ifq} = Q_q \qquad \forall q \in \mathcal{Q}, \tag{1.120}$$

$$z_{ifq} \leq Q_{iq} y_{if} \qquad \forall i \in \mathcal{V}\setminus\{0\}, q \in \mathcal{Q}, f \in \mathcal{F}, \tag{1.121}$$

$$\sum_{i\in V\setminus\{0\}}\sum_{q\in\mathcal{Q}} z_{ifq} \leq Q \qquad \forall f \in \mathcal{F}, \tag{1.122}$$

$$u_i - u_j + N\sum_{f\in\mathcal{F}} x_{ijf} \leq N-1 \qquad \forall i,j \in \mathcal{V}\setminus\{0\}, i \neq j, \tag{1.123}$$

$$x_{ijf} \in \{0,1\} \qquad \forall i,j \in \mathcal{V}, i \neq j, f \in \mathcal{F}, \tag{1.124}$$

$$y_{if} \in \{0,1\} \qquad \forall i \in \mathcal{V}, f \in \mathcal{F}, \tag{1.125}$$

$$z_{ifq} \geq 0 \qquad \forall i \in \mathcal{V}\setminus\{0\}, q \in \mathcal{Q}, f \in \mathcal{F}, \tag{1.126}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V}\setminus\{0\}. \tag{1.127}$$

The objective function (1.116) is to minimize the traveling and purchasing costs. Constraints (1.117) and Constraints (1.118) are flow conservation constraints and impose that an arc enters and exits each selected vertex. Constraints (1.119) ensure that every market can be visited at most by one vehicle. Constraints (1.120) ensure that exactly the quantity of products required shall be purchased. Constraints (1.121) impose that the quantity of a product purchased by a vehicle at a market is less than the available quantity at this market. Constraints (1.122) guarantee that the capacity of the vehicle is respected. Constraints (1.123) are subtour elimination constraints. Constraints (1.124) – (1.127) define the variables.

The interested reader is referred to the survey on the TPP and its variants by Manerba *et al.* (2017) that to the best of our knowledge covers the entire literature up to now.

1.5 Conclusions and perspectives

Routing problems are usually defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the set of vertices. Classical routing problems require to visit every vertex of the graph. Several articles and books survey the literature extensively (see for example Laporte (2009); Toth & Vigo (2014)). On the other side, when not all the vertices are required to visit, only one paper by Laporte & Martín (2007) presents a wide overview. However, this paper focuses on works where a single resource is available to perform deliveries.

In this survey, we have presented several non-HRP problems including both the single-vehicle case and the multi-vehicle case. Throughout the survey, for each of these problems, we presented its definition, a compact mathematical formulation, we provided a literature review and gave some of its applications.

Among the problems presented in this paper, those that have been largely studied and still draw a lot of attention by scholars are the TSPPs, the VRPPs and their variants, especially the OP, TOP and their variants. Most of the works on the CTP and mCTP deal with variants instead of the basic problems. The RSP and CmRSP are more popular compared with the MCP and, in recent years, researchers got interested in the bi-objective variants of the RSP and the multi-depot version of the CmRSP. The GTSP and GVRP start to gain more and more attention in recent years, and there is still considerable room for research on the variants of these problems.

With regard to the solution approaches, in general, the most efficient exact solution methodology seems to be branch-and-cut algorithms for the single-vehicle case and branch-and-price methods for the multi-vehicle case. Efficient heuristics are often metaheuristics with hybrid strategies combining several heuristics, e.g., variable neighborhood search, large neighborhood search, iterated local search, greedy randomized adaptive search procedure, genetic algorithm, simulated annealing.

The non-HRP problems have a wide and variety of applications in transportation and telecommunications, for example, the tourist trip design problem, humanitarian relief, and city telecommunication network design.

A particular mention is given to new delivery systems. Due to the quick development of the e-commerce and the popularity of online shopping, more and more packages need to be delivered to customers, mainly located in urban areas. Delivery companies are seeking creative and innovative ways to efficiently make the delivery. In recent years, some new delivery services appear, such as, delivering a customer's package to his/her home, a pick-up point like a locker, or to the customer's car. If all these delivery services are combined together, customers are able to provide several different locations to receive packages. The courier only needs to deliver the package to one of these locations. Such logistics problems are non-HRP since they do not require visits of all the customer locations. They are of interest because they might increase the rate of successful first-time deliveries and decrease the delivery costs.

Chapter 2

Mixed integer programming formulations for the GTSPTW

Contents

This chapter corresponds to a working paper "Mixed integer programming formulations for the generalized traveling salesman problem with time windows", to be submitted soon. The Section 2.2 corresponds to the paper "A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for routing problems with time windows", submitted to Operation Research Letters on 5 August, 2019.

Abstract: The generalized traveling salesman problem with time windows (GTSPTW) is defined on a directed graph where the vertex set is partitioned into clusters. One cluster contains only the depot. Each vertex is associated with a time window, during which the visit must take place if the vertex is visited. The objective is to find a minimum cost tour starting and ending at the depot such that each cluster is visited exactly once and time constraints are respected, i.e., for each cluster, a single vertex is visited during its time window. In this paper, four mixed integer linear programming formulations for the GTSPTW are proposed and compared. They are based on different definitions of variables. All the formulations are compact, which means the number of decision variables and constraints is polynomial with respect to the size of the instance. We establish theoretically the dominance relations between their linear relaxations. We also conduct experimental results to compare the linear relaxations and branch-and-bound performances for the four formulations. The results show that two formulations are better than the other ones.

2.1 Introduction

The problem addressed in this paper is the generalized traveling salesman problem with time windows (GTSPTW), where clusters represent possible delivery locations associated with a customer. In this paper, we propose and compare four mixed integer programming (MIP) formulations for this problem, based on different definitions of variables.

The GTSPTW can be formally defined as follows: given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, the set of vertices $\mathcal{V} = \{0, 1, ..., N\}$ is partitioned into $\mathcal{C}_0 = \{0\}, \mathcal{C}_1, ..., \mathcal{C}_K$ clusters. $\mathcal{K} = \{0, 1, ..., K\}$ denotes the cluster index set. Cluster $\mathcal{C}_0$ contains only the depot. $\mathcal{C}_k$, $k > 0, k \in \mathcal{K}$ represents the set of alternative locations on which customer k can be delivered. Each vertex is associated with a TW $[E_i, L_i], i \in \{0, 1, ..., N\}$ with $[E_0, L_0] = [0, T]$ representing the optimization time horizon. If a vertex is visited, the visit occurs during its TW. An early arrival leads to waiting time while a late arrival causes infeasibility. Arcs are only defined between vertices belonging to different clusters, that is, $\mathcal{A} = \{(i, j) | i \in \mathcal{C}_k, j \in \mathcal{C}_l, k \neq l, k, l \in \mathcal{K}\}$. Each arc $(i, j) \in \mathcal{A}$ is associated with a traveling cost C_{ij} and a positive traveling time $T_{ij} \geq 0$. The objective of the GTSPTW is to find a minimum cost tour starting and ending at the

depot such that each cluster is visited exactly once and time constraints are respected, i.e., one vertex of each cluster is visited during its time window.

A straightforward application for the GTSPTW is the routing problem in the context of last mile delivery including multiple delivery services. The most common delivery option is home/workplace delivery. Customers wait at home/workplace to get their packages. Besides, the delivery can be made to pick-up points such as dedicated lockers or stores. In this case, customers can retrieve their packages after delivery has been performed. Moreover, in recent years, a new concept called *trunk/in-car delivery*, has been proposed. Here, customers' packages can be delivered to the trunks of cars (Hawkins, 2018; Kirsten, 2016). All these delivery services can be combined, and instead of choosing one delivery location during the online purchase, a customer can propose a set of delivery locations (home/workplace, pick-up points, and car trunk) with the associated timing constraints. The last mile delivery system with multiple delivery services provides customers more options and flexibility considering their own convenience. Moreover, it might increase the rate of successful first-time deliveries and decrease delivery costs. In this work, the GTSPTW is the problem encountered for the one vehicle case: clusters represent possible delivery locations associated with a customer, and it is assumed that a single vehicle is able to deliver all the customers on the same tour. In Figure 2.1, we provide an example of the routing problem considered in the context of the last mile delivery with multiple delivery services. Four customers are represented with their associated locations grouped into a dotted circle, representing the different *clusters*. Every possible delivery location is associated with a time window (TW) during which delivery should occur. In the case of a home or trunk delivery, the TW represents the period when the customer or the customer's car is present at that location. In the case of a pick-up point, the TW represents the period during which the courier can deliver the package before the customer arrives at the location and picks it up. One solution consists in selecting one location for each customer and the sequence of visits such that the delivery at each selected location takes place during its TW.

The GTSPTW is related to other routing problems. When TWs are not considered, the GTSPTW reduces to the generalized traveling salesman problem (GTSP) (Fischetti *et al.*, 1997; Kara *et al.*, 2012; Pop, 2007). Fischetti *et al.* (1997) proposed an efficient branch-and-cut algorithm to solve the symmetric version of the GTSP. They developed exact and heuristic separation procedures for some classes of facet-defining inequalities.

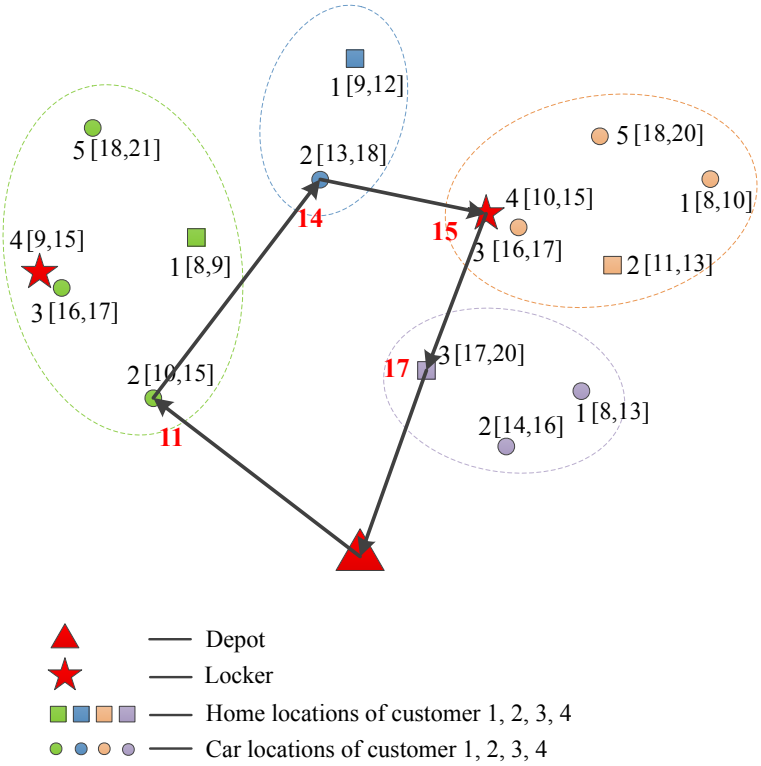


Figure 2.1: An example of GTSPTW instance.

Kara *et al.* (2012) proposed two formulations for the GTSP with polynomial size with respect to the number of nodes.

When all clusters contain only one vertex, the GTSPTW reduces to the traveling salesman problem with time windows (TSPTW) (Ascheuer *et al.*, 2001; Dash *et al.*, 2012). Ascheuer *et al.* (2001) proposed several formulations for the asymmetric TSPTW and compared them within a branch-and-cut scheme. Dash *et al.* (2012) presented a formulation for the TSPTW based on the partitioning of the TW into sub-windows.

To the best of our knowledge, there is no work that has been done for the GTSPTW. There are some works related to the multi-vehicle case. The special case where TWs of locations associated with the same customer do not overlap is called the vehicle routing problem with roaming delivery locations (VRPRDL) (Ozbaygin *et al.*, 2017; Reyes *et al.*, 2017), inspired by the trunk delivery. The VRPRDL consists of finding a minimum-cost set of routes for a fleet of capacitated vehicles in which the order of a customer has to be delivered to the trunk of the customer's car when the car is parked. The TWs of locations within a cluster have a specific structure since they are

non-overlapping. Reyes *et al.* (2017) developed a construction heuristic based on a greedy randomized adaptive search procedure, and an improvement heuristic based on an ALNS. Ozbaygin *et al.* (2017) developed a branch-and-price algorithm.

The contributions of this paper are the following. First, we provide four mixed integer programming formulations for the GTSPTW, based on decisions variables related to vertices or clusters. Second, we provide theoretical results on the dominance of the linear programming (LP) relaxations of the proposed formulations. Finally, the formulations are experimentally compared with two sets of instances derived from the GTSP.

The remainder of this paper is organized as follows. Since the proposed four formulations use the Miller-Tucker-Zemlin (MTZ) (Miller *et al.*, 1960) form of subtour elimination constraints, before presenting the formulations, lifted MTZ constraints for the routing problems with time windows are proposed in Section 2.2. The four mathematical formulations are provided in Section 2.3. Section 2.4 presents the theoretical dominance between the LP relaxations of the formulations. Section 2.5 reports the computational results. Finally, conclusions are drawn in Section 2.6.

2.2 Lifting MTZ subtour elimination constraints for routing problems with time windows

The traveling salesman problem with time windows (TSPTW) is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0, 1, ..., N\}$ is the vertex set and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is the arc set. 0 is the depot, from where the salesman starts and ends the visiting tour. Each vertex $i \in \mathcal{V} \setminus \{0\}$ must be visited within a specified time window $[a_i, b_i]$ and waiting time is allowed, i.e., the salesman can arrive at i before a_i and wait until a_i to visit vertex i. The time window of the depot is $[a_0, b_0]$: the salesman leaves the depot after a_0 and returns to the depot before b_0. Each arc (i, j) is associated with a travel cost $c_{ij} \geq 0$ and a travel time $t_{ij} \geq 0$. The TSPTW consists of determining a tour such that the total travel cost is minimized and every vertex $i \in \mathcal{V}$ is visited exactly once within its time window $[a_i, b_i]$ (Dumas *et al.* (1995), Gendreau *et al.* (1998)).

The TSPTW can be formulated using two types of variables. x_{ij} is a binary variable that equals to 1 if and only if arc $(i, j) \in \mathcal{A}$ is used in the solution. u_i specifies the service start time at vertex $i \in \mathcal{V} \setminus \{0\}$, and u_0 is the departure time from the depot.

The TSPTW can be formulated as follows:

$$\text{minimize} \quad \sum_{i \in \mathcal{V}} \sum_{\substack{j \in \mathcal{V} \\ j \neq i}} c_{ij} x_{ij} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{\substack{j \in \mathcal{V} \\ j \neq i}} x_{ij} = 1 \qquad \forall i \in \mathcal{V}, \tag{2.2}$$

$$\sum_{\substack{i \in \mathcal{V} \\ j \neq i}} x_{ij} = 1 \qquad \forall j \in \mathcal{V}, \tag{2.3}$$

$$u_i - u_j + M x_{ij} \leq M - t_{ij} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{0\}, i \neq j, \tag{2.4}$$

$$a_i \leq u_i \leq b_i \qquad \forall i \in \mathcal{V}, \tag{2.5}$$

$$u_i + t_{i0} x_{i0} \leq b_0 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j, \tag{2.7}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{V}. \tag{2.8}$$

The formulation adapts the formulation proposed by Dantzig *et al.* (1954) for the traveling salesman problem where the subtour elimination constraints are written in a MTZ fashion. The objective function (2.1) minimizes the total cost. Constraints (2.2) ensure that each vertex is visited exactly once. Constraints (2.3) are flow conservation constraints. Constraints (2.4) guarantee the feasibility of the tour with respect to time constraints. These constraints are also known as MTZ subtour elimination constraints since they ensure that the solution does not contain subtours disconnected from the depot. M is a large constant such that $M \geq \max_{i,j \in \mathcal{V}} \{b_i - a_j + t_{ij}\}$. Constraints (2.5) ensure that each vertex is visited during its time window, and the salesman leaves the depot during its time window. Constraints (2.6) ensure that the salesman is back at the depot before b_0. Constraints (2.7) and (2.8) define the variables.

Here we discuss on the lifting of the Constraints (2.4) for the TSPTW. In the paper by Desrochers & Laporte (1991), the authors proposed a lifted version of these constraints. Because of time window constraints, some arcs (i, j) are *infeasible*. Indeed, if $b_j < a_i + t_{ij}$, then it is not possible to visit vertex j right after visiting vertex i. According to Desrochers & Laporte (1991), if arcs (i, j) and (j, i) are both *feasible*, by taking into account the inverse arcs (j, i), Constraints (2.4) can be strengthened as:

$$u_i - u_j + M x_{ij} + (M - t_{ij} + \min\{-t_{ji}, b_j - a_i\}) x_{ji} \leq M - t_{ij} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{0\}, i \neq j. \tag{2.9}$$

However, we find that these valid inequalities are incorrectly written since they can eliminate feasible or optimal solutions.

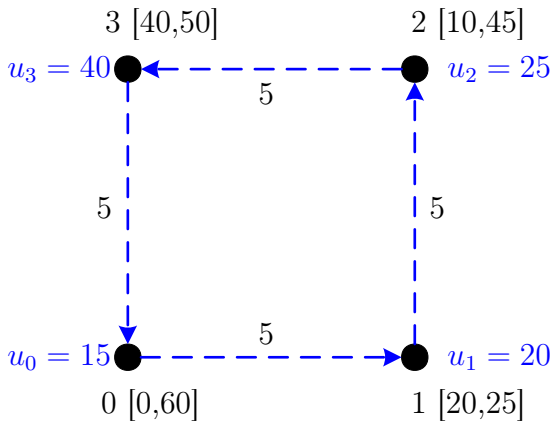


Figure 2.2: Example where the optimal solution is cut off by Constraints (2.9).

First, we provide an example, depicted in Figure 2.2, to illustrate the situation. Let us consider an instance with four vertices $0, 1, 2$, and 3 in the graph and vertex 0 represents the depot. Vertices $0, 1, 2$, and 3 are associated with time windows $[0, 60]$, $[20, 25]$, $[10, 45]$, and $[40, 50]$ respectively. Vertices $0, 1, 2$, and 3 are located at coordinates $(0, 0)$, $(5, 0)$, $(5, 5)$ and $(0, 5)$ respectively. For each arc (i, j), we consider that c_{ij} and t_{ij} are equal to the Euclidean distance between i and j. It is clear that the tour $0 - 1 - 2 - 3 - 0$ is feasible and optimal. Feasible values for u variables are: $u_0 = 15, u_1 = 20, u_2 = 25, u_3 = 40$.

When $i = 2$ and $j = 1$, Constraints (2.9) are:

$$u_2 - u_1 + Mx_{21} + (M - t_{21} + \min\{-t_{12}, b_1 - a_2\})x_{12} \leq M - t_{21}.$$

Thus, when Constraints (2.9) for $i = 2$ and $j = 1$ are applied to this solution they will provide the following equation:

$$u_2 - u_1 + 0 + (M - 5 + \min\{-5, 25 - 10\}) \leq M - 5;$$

$$u_2 \leq u_1 + 5.$$

Thus, the constraint $u_2 \leq u_1 + 5$ is imposed. Similarly, when $i = 3$ and $j = 2$, Constraints (2.9) are $u_3 \leq u_2 + 5$. These two constraints lead to $u_3 \leq u_1 + 10$. However, $u_1 \leq 25$ and $u_3 \geq 40$. Hence, route $0 - 1 - 2 - 3 - 0$ is not feasible when considering Constraints (2.9).

Therefore, we correct the Constraints (2.9) as follows.

Proposition 2.1. *The constraints*

$$u_i - u_j + Mx_{ij} + (M - t_{ij} + a_j - b_i)x_{ji} \leq M - t_{ij} \quad \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{0\}, i \neq j \quad (2.10)$$

are valid inequalities for the TSPTW.

Proof. Consider the general constraints:

$$u_i - u_j + Mx_{ij} + (M - t_{ij} + \alpha_{ji})x_{ji} \leq M - t_{ij} \quad \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{0\}, i \neq j. \quad (2.11)$$

Given $i \in \mathcal{V}, j \in \mathcal{V} \setminus \{0\}, i \neq j$, we seek for the largest value of α_{ji} such that the corresponding Constraint (2.11) is valid.

If $x_{ji} = 0$, these constraints are obviously satisfied for any value of α_{ji}. If $x_{ji} = 1$, (vertex i is visited right after visiting vertex j), then $x_{ij} = 0$, and we obtain the following constraints:

$$u_i - u_j + \alpha_{ji} \leq 0. \quad (2.12)$$

i.e.,

$$\alpha_{ji} \leq u_j - u_i. \quad (2.13)$$

We are thus seeking for the largest possible value of α_{ji} such that Constraint (2.13) is valid for any feasible values of decision variables u_j and u_i. From Constraints (2.5), we have: $u_i \leq b_i$ and $u_j \geq a_j$. Thus, we obtain:

$$a_j - b_i \leq u_j - u_i. \quad (2.14)$$

This means that α_{ji} cannot be bigger than $a_j - b_i$. Otherwise Constraint (2.13) is violated when $u_j = a_j$ and $u_i = b_i$. Hence, we set $\alpha_{ji} = a_j - b_i$. □

Let us continue to consider the example discussed above in Figure 2.2. When $i = 2$ and $j = 1$, Constraints (2.10) impose $u_2 \leq u_1 + 25$. When $i = 3$ and $j = 2$, Constraints (2.10) impose $u_3 \leq u_2 + 40$. These two constraints lead to $u_3 \leq u_1 + 65$, which means route $0 - 1 - 2 - 3 - 0$ is feasible.

Moreover, it can be noticed that when the optimal solution contains waiting times, the formulation (2.1)∼(2.8) may have multiple optimal solutions because of the timing variables u_i. Indeed, given optimal values for the x_{ij} variables, there may be several values for the u_i variables that satisfy Constraints (2.4) and (2.5). Moreover, given optimal values for the x_{ij} variables, there are always feasible values for the u_i variables such that each vertex (except the depot) is visited as early as possible, namely minimizing waiting times (see for example the solution in Figure 2.2).

Thus, in the following, we propose supervalid inequalities for the TSPTW. An inequality is *supervalid* if it does not cut off all optimal solutions. This concept is a generalization of the concept of valid inequalities and has been introduced by Israeli & Wood (2002).

Proposition 2.2. *The constraints*

$$u_i - u_j + Mx_{ij} + (M - t_{ij} + \min\{-t_{ji}, a_j - a_i\})x_{ji} \leq M - t_{ij} \quad \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{0\}, i \neq j \tag{2.15}$$

are supervalid inequalities for the TSPTW.

Proof. We seek for a value of α_{ji} such that Constraints (2.11) are supervalid, i.e., they are valid for at least one optimal solution. If $x_{ji} = 0$, these constraints are satisfied for any value of α_{ji}. If $x_{ji} = 1$(vertex i is visited right after visiting vertex j), then $x_{ij} = 0$, and we obtain Constraints (2.13). Thus, we seek for a value of α_{ji} such that Constraints (2.13) are valid for the values of the decision variables u_j and u_i for at least one optimal solution.

In order to provide such a value, we consider an optimal solution where each vertex is visited as soon as possible.

Two cases may be considered. In the first case, $a_j + t_{ji} \geq a_i$. This means that vertex i can be visited right after vertex j without any waiting time. Hence, we have $u_i = u_j + t_{ji}$. We then obtain:

$$-t_{ji} = u_j - u_i \tag{2.16}$$

Hence, $\alpha_{ji} = -t_{ji}$ is valid for this first case.

In the second case, $a_j + t_{ji} < a_i$. This means that a waiting time may be required before visiting vertex i. Let us suppose that the value of u_j has been fixed. The value of u_i will then be chosen such that vertex i is visited as soon as possible. If $u_j + t_{ji} \geq a_i$, then i can be visited right after j without waiting time. Then, similarly to the first case, $\alpha_{ji} = -t_{ji}$ is valid. If $u_j + t_{ji} < a_i$, then a waiting time is required, and vertex i is visited as soon as possible, i.e. $u_i = a_i$. From Constraints (2.5), we have $u_j \geq a_j$. We then obtain:

$$a_j - a_i \leq u_j - u_i \tag{2.17}$$

Hence, when $\alpha_{ji} = a_j - a_i$, Constraints (2.13) are valid in this case.

In all cases, we can set $\alpha_{ji} = \min\{-t_{ji}, a_j - a_i\}$.

Using the example introduced previously and depicted in Figure 2.2, it can be observed that Constraints (2.15) do not cut the solution proposed in Figure 2.2. However,

other optimal solutions are eliminated. Let us consider another solution s' with the same values for x_{ij} variables and $u_0 = 15, u_1 = 20, u_2 = 35$, and $u_3 = 40$. This solution is valid and optimal with respect to the formulation of the TSPTW. However, Constraints (2.15) applied to $i = 2$ and $j = 1$ give $u_2 - u_1 + 0 + (M - 5 + \min\{-5, 20 - 10\}) \leq M - 5$. Thus the constraint is $u_2 \leq u_1 + 5$, and the solution s' would be cut off by Constraints (2.15).

Thus, Constraints (2.15) are supervalid inequalities for the TSPTW. □

Note that this result can be extended to routing problems with time windows such as the Generalized TSPTW (Yuan *et al.* (2019a)), the Vehicle Routing Problem with TW (VRPTW) (Pecin *et al.* (2017)), the Generalized VRPTW (Yuan *et al.* (2019c)).

2.3 Four compact formulations for the GTSPTW

In order to provide compact formulations for the GTSPTW, we use some additional notations. Let $\mathcal{A}_c = \{(h, k) | h, k \in \mathcal{K}, h \neq k\}$ be the set of ordered pairs of clusters. For any node $i \in \mathcal{V}$, we define $\delta^+(i) = \{(i, j) \in \mathcal{A} | j \in \mathcal{V}\}$ the set of arcs leaving vertex i, and $\delta^-(i) = \{(j, i) \in \mathcal{A} | j \in \mathcal{V}\}$ the set of arcs entering vertex i. Similarly, for any cluster $h \in \mathcal{K}$, we define $\delta_c^+(h) = \{(h, k) \in \mathcal{A}_c | k \in \mathcal{K}\}$, and $\delta_c^-(h) = \{(k, h) \in \mathcal{A}_c | k \in \mathcal{K}\}$.

The first model is based on the classical variables used to model the GTSP and the TSPTW. We define three sets of variables:

- $x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ belongs to the tour,} \\ 0 & \text{otherwise,} \end{cases}$ $\forall (i, j) \in \mathcal{A}$;
- $y_i = \begin{cases} 1 & \text{if vertex } i \text{ is visited in the tour,} \\ 0 & \text{otherwise,} \end{cases}$ $\forall i \in \mathcal{V}$;
- $t_i \geq 0$ is the service time at vertex i, $\forall i \in \mathcal{V}$.

Note that, for the depot, the service time t_0 represents the departure time, and if a vertex $i \in \mathcal{V}$ is not visited, the corresponding variable t_i will be 0.

Using these variables, a first mathematical programming formulation $\mathcal{F}1$ is:

$$(\mathcal{F}1) \quad \text{minimize} \quad \sum_{(i,j) \in \mathcal{A}} C_{ij} x_{ij} \tag{2.18}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{C}_k} y_i = 1 \qquad \forall k \in \mathcal{K}, \tag{2.19}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V}, \tag{2.20}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i \qquad \forall i \in \mathcal{V}, \qquad (2.21)$$

$$E_i y_i \leq t_i \leq L_i y_i \qquad \forall i \in \mathcal{V}, \qquad (2.22)$$

$$t_i - t_j + T_{ij}x_{ij} \leq L_i y_i - E_j y_j - (L_i - E_j)x_{ij} \qquad \forall (i,j) \in \mathcal{A}, j \neq 0, \qquad (2.23)$$

$$t_i + T_{i0}x_{i0} \leq L_0 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \qquad (2.24)$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{V}, \qquad (2.25)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, \qquad (2.26)$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{V}. \qquad (2.27)$$

The objective function (2.18) minimizes the overall traveling cost. Constraints (2.19) ensure that exactly one vertex from each cluster is visited. Constraints (2.20) and (2.21) ensure flow conservation between the visited vertices, and that no flow is passing through the unvisited vertices. Constraints (2.22) ensure that a vertex is visited during its TW. Constraints (2.23) ensure that the service times are consistent, i.e., if vertex j is visited just after vertex i, then (2.23) will ensure $t_j \geq t_i + T_{ij}$. These constraints are MTZ form of subtour elimination constraints. Constraints (2.24) make sure that the return to the depot occurs before the end of its TW. Constraints (2.25)$\sim$(2.27) are variable definitions.

Moreover, we can extend the supervalid inequalities proposed in Section 2.2 for the MTZ constraints (2.23). Because of time window constraints, some arcs (i,j) are *infeasible*. Indeed, if $E_i + T_{ij} > L_j$, then it is not possible to visit vertex j right after visiting vertex i. If arcs (i,j) and (j,i) are both *feasible*, by taking into account the inverse arcs (j,i), the MTZ constraints (2.23) in $\mathcal{F}1$ can be replaced by the following supervalid inequalities:

$$t_i - t_j + T_{ij}x_{ij} + min\{-T_{ji}, E_j - E_i\}x_{ji} \leq L_i y_i - E_j y_j - (L_i - E_j)(x_{ij} + x_{ji}). \qquad (2.28)$$

A second formulation is based on the following observation. Since only one vertex is selected in each cluster, we can define a single time variable per cluster instead of defining a time variable for every vertex as proposed in formulation $\mathcal{F}1$. Hence, instead of using variables $t_i, \forall i \in \mathcal{V}$, we define a new set of time variables:

- $\tau_k \geq 0$ is the service time at cluster k, $\forall k \in \mathcal{K}$.

Using these time variables, we have a second formulation $\mathcal{F}2$. The objective function and constraints (2.19)$\sim$(2.21), (2.25)$\sim$(2.26) are the same as in $\mathcal{F}1$. Constraints

(2.22)∼(2.24) are replaced by (2.29)∼(2.31) respectively.

$(\mathcal{F}2)$ minimize (2.18)

s.t. (2.19) ∼ (2.21)

$$\sum_{i\in\mathcal{C}_k} E_i y_i \le \tau_k \le \sum_{i\in\mathcal{C}_k} L_i y_i \qquad \forall k \in \mathcal{K}, \quad (2.29)$$

$$\tau_h - \tau_k + \sum_{\substack{i\in\mathcal{C}_h \\ j\in\mathcal{C}_k}} T_{ij}x_{ij} \le \sum_{i\in\mathcal{C}_h} L_i y_i - \sum_{j\in\mathcal{C}_k} E_j y_j - \sum_{\substack{i\in\mathcal{C}_h \\ j\in\mathcal{C}_k}} (L_i - E_j)x_{ij} \quad \forall h \in \mathcal{K},\ k \in \mathcal{K}\setminus\{0\}, h \ne k, \quad (2.30)$$

$$\tau_k + \sum_{i\in\mathcal{C}_k} T_{i0}x_{i0} \le L_0 \qquad \forall k \in \mathcal{K}\setminus\{0\}, \quad (2.31)$$

(2.25) ∼ (2.26)

$$\tau_k \ge 0 \qquad \forall k \in \mathcal{K}. \quad (2.32)$$

Constraints (2.29) ensure that each cluster is visited during the TW of its corresponding visited vertex. Constraints (2.30) ensure that the service times between clusters are consistent. They are MTZ type of constraints and can eliminate subtours. Constraints (2.31) make sure that the return to the depot occurs before its TW closes. Constraints (2.32) are variable definitions.

For this second formulation, it is also possible to replace the MTZ constraints (2.30) in $\mathcal{F}2$ by the following supervalid inequalities:

$$\tau_h - \tau_k + \sum_{\substack{i\in\mathcal{C}_h \\ j\in\mathcal{C}_k}} T_{ij}x_{ij} + \sum_{\substack{i\in\mathcal{C}_h \\ j\in\mathcal{C}_k}} min\{-T_{ji}, E_j - E_i\}x_{ji} \le \sum_{i\in\mathcal{C}_h} L_i y_i - \sum_{j\in\mathcal{C}_k} E_j y_j - \sum_{\substack{i\in\mathcal{C}_h \\ j\in\mathcal{C}_k}} (L_i - E_j)(x_{ij} + x_{ji}). \quad (2.33)$$

A third formulation is proposed based on the next observation. At most one arc between two clusters is chosen to connect two clusters. Thus, instead of defining arc variables between two vertices belonging to two different clusters as in $\mathcal{F}1$ and $\mathcal{F}2$, we can define an arc variable between two clusters. In this case, we also need variables to represent the time and the cost to travel from a cluster $h \in \mathcal{K}$ to another cluster $k \in \mathcal{K}$. Instead of using variables $x_{ij}, \forall (i,j) \in \mathcal{A}$, we define new sets of variables:

- $\chi_{hk} = \begin{cases} 1 & \text{if the tour visits cluster } h \text{ just before cluster } k, \\ 0 & \text{otherwise,} \end{cases}$ $\forall (h,k) \in \mathcal{A}_c$;
- $\tau_{hk} \ge 0$ is the traveling time from cluster h to cluster k, $\forall (h,k) \in \mathcal{A}_c$;

- $\varsigma_{hk} \geq 0$ is the traveling cost from cluster h to cluster k, $\forall (h,k) \in \mathcal{A}_c$.

Note that if the tour does not visit cluster h just before cluster k ($\chi_{hk} = 0$), then the corresponding variables τ_{hk} and ς_{hk} will be 0.

Using these new sets of variables in addition to binary variables y_i and time variables t_i as in $\mathcal{F}1$, a third formulation $\mathcal{F}3$ can be defined as follows:

$$(\mathcal{F}3) \quad minimize \quad \sum_{(h,k)\in\mathcal{A}_c} \varsigma_{hk} \tag{2.34}$$

$$s.t. \quad (2.19)$$

$$\sum_{(h,k)\in\delta_c^+(h)} \chi_{hk} = 1 \qquad \forall h \in \mathcal{K}, \tag{2.35}$$

$$\sum_{(k,h)\in\delta_c^-(h)} \chi_{kh} = 1 \qquad \forall h \in \mathcal{K}, \tag{2.36}$$

$$(\chi_{hk} + y_i + y_j - 2)C_{ij} \leq \varsigma_{hk} \qquad \forall (h,k) \in \mathcal{A}_c, i \in \mathcal{C}_h, j \in \mathcal{C}_k, \tag{2.37}$$

$$C_{hk}^{min}\chi_{hk} \leq \varsigma_{hk} \leq C_{hk}^{max}\chi_{hk} \qquad \forall (h,k) \in \mathcal{A}_c, \tag{2.38}$$

$$(\chi_{hk} + y_i + y_j - 2)T_{ij} \leq \tau_{hk} \qquad \forall (h,k) \in \mathcal{A}_c, i \in \mathcal{C}_h, j \in \mathcal{C}_k, \tag{2.39}$$

$$T_{hk}^{min}\chi_{hk} \leq \tau_{hk} \leq T_{hk}^{max}\chi_{hk} \qquad \forall (h,k) \in \mathcal{A}_c, \tag{2.40}$$

$$(2.22)$$

$$\sum_{i\in\mathcal{C}_h} t_i - \sum_{j\in\mathcal{C}_k} t_j + \tau_{hk} \leq M_{hk}(1 - \chi_{hk}) \qquad \forall (h,k) \in \mathcal{A}_c, k \neq 0, \tag{2.41}$$

$$\sum_{i\in\mathcal{C}_k} t_i + \tau_{k0} \leq L_0 \qquad \forall k \in \mathcal{K} \setminus \{0\}, \tag{2.42}$$

$$(2.25), (2.27)$$

$$\chi_{hk} \in \{0,1\} \qquad \forall (h,k) \in \mathcal{A}_c, \tag{2.43}$$

$$\tau_{hk} \geq 0 \qquad \forall (h,k) \in \mathcal{A}_c, \tag{2.44}$$

$$\varsigma_{hk} \geq 0 \qquad \forall (h,k) \in \mathcal{A}_c; \tag{2.45}$$

where $C_{hk}^{min} = \min_{i\in\mathcal{C}_h, j\in\mathcal{C}_k}\{C_{ij}\}$, $C_{hk}^{max} = \max_{i\in\mathcal{C}_h, j\in\mathcal{C}_k}\{C_{ij}\}$, $T_{hk}^{min} = \min_{i\in\mathcal{C}_h, j\in\mathcal{C}_k}\{T_{ij}\}$, $T_{hk}^{max} = \max_{i\in\mathcal{C}_h, j\in\mathcal{C}_k}\{T_{ij}\}$, $M_{hk} = L_h^{max} - E_k^{min}$, with $L_h^{max} = \max_{i\in\mathcal{C}_h}\{L_i\}$, and $E_k^{min} = \min_{j\in\mathcal{C}_k}\{E_j\}$.

The objective function (2.34) minimizes the overall traveling cost. Constraints (2.35) and (2.36) are flow conservation constraints. Constraints (2.37) and (2.38) provide bounds on variables ς_{hk}, while constraints (2.39) and (2.40) provide bounds on variables τ_{hk}. Constraints (2.41) ensure that the service times are consistent, i.e., if vertex

$j \in \mathcal{C}_k$ is visited just after vertex $i \in \mathcal{C}_h$, then (2.41) will ensure $t_j \geq t_i + \tau_{hk}$. Constraints (2.42) ensure that the return to the depot occurs before the end of its TW. Constraints (2.43)$\sim$(2.45) are variable definitions.

Note that if for every arc $(i,j) \in \mathcal{A}$, the traveling time T_{ij} equals the traveling cost C_{ij}, then Constraints (2.37)$\sim$(2.38) on one hand, and Constraints (2.39)$\sim$(2.40) on the other hand are the same.

Finally, the observation made to propose formulation $\mathcal{F}2$ is still valid: only one vertex is selected in each cluster, so we can define a single time variable per cluster instead of defining a time variable for every vertex. We propose a formulation $\mathcal{F}4$, derived from formulation $\mathcal{F}3$ using time variables $\tau_k, \forall k \in \mathcal{K}$ instead of time variables $t_i, \forall i \in \mathcal{V}$.

$$
\begin{aligned}
(\mathcal{F}4) \quad & \text{minimize} && (2.34) \\
s.t. \quad & && (2.19) \\
& && (2.35) \sim (2.40) \\
& && (2.29) \\
& \tau_h - \tau_k + \tau_{hk} \leq M_{hk}(1 - \chi_{hk}) && \forall (h,k) \in \mathcal{A}_c, k \neq 0, && (2.46) \\
& \tau_k + \tau_{k0} \leq L_0 && \forall k \in \mathcal{K} \setminus \{0\}. && (2.47) \\
& && (2.25), (2.32), (2.43) \sim (2.45)
\end{aligned}
$$

Constraints (2.46) ensure that the service times between clusters are consistent, and Constraints (2.47) ensure that the return to the depot occurs before the end of its TW.

Notice that all the proposed formulations use MTZ type of constraints to eliminate subtours. If an instance contains two vertices $i, j \in \mathcal{V}$ such that $T_{ij} = T_{ji} = 0$ and their TWs intersect ($[E_i, L_i] \cap [E_j, L_j] \neq \emptyset$), then the MTZ constraints cannot eliminate the subtour between vertices i and j since they can be visited at the same time. In such a case, the MTZ constraints can be replaced by classical subtour elimination constraints for the subsets of vertices that can be visited at the same time. In the remainder of this paper, we assume that any two vertices i, j cannot be visited at the same time, i.e., $[E_i, L_i] \cap [E_j, L_j] \neq \emptyset$ and $T_{ij} = T_{ji} = 0$ cannot be simultaneously true. We focus on the proposed formulations using only MTZ subtour elimination constraints.

2.4 Dominance between formulations

Let $LR(\mathcal{F}1)$, $LR(\mathcal{F}2)$, $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$ denote the LP relaxations of $\mathcal{F}1$, $\mathcal{F}2$, $\mathcal{F}3$ and $\mathcal{F}4$ respectively. This mean that Constraints (2.25), (2.26) and (2.43) are respectively replaced by:

$$0 \leq y_i \leq 1 \qquad \forall i \in \mathcal{V}, \tag{2.48}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall (i,j) \in \mathcal{A}, \tag{2.49}$$

$$0 \leq \chi_{hk} \leq 1 \qquad \forall (h,k) \in \mathcal{A}_c. \tag{2.50}$$

In this section, we provide some theoretical dominance relationships on the LP relaxations of the proposed formulations.

Proposition 2.3. *Formulations $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$ are equivalent.*

Proof. 1) First, we demonstrate that a feasible solution of $LR(\mathcal{F}3)$ is feasible for $LR(\mathcal{F}4)$.

Let us consider a feasible solution of $LR(\mathcal{F}3)$. In order to show that this solution is feasible for $LR(\mathcal{F}4)$, we need to verify that it respects all the constraints of $LR(\mathcal{F}4)$. By comparing $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$, we see that only Constraints (2.29), (2.46) and (2.47) in $LR(\mathcal{F}4)$ have to be verified.

For $LR(\mathcal{F}4)$, we define the time variables τ_k as follows:

$$\tau_k = \sum_{i \in \mathcal{C}_k} t_i \qquad \forall k \in \mathcal{K}. \tag{2.51}$$

Given a cluster $k \in \mathcal{K}$, by summing up Constraints (2.22): $E_i y_i \leq t_i \leq L_i y_i, \forall i \in \mathcal{V}$ in $LR(\mathcal{F}3)$ over all vertices in $\mathcal{C}_k$, we have $\sum_{i \in \mathcal{C}_k} E_i y_i \leq \sum_{i \in \mathcal{C}_k} t_i \leq \sum_{i \in \mathcal{C}_k} L_i y_i, \forall k \in \mathcal{K}$. Then, according to the definition of variable τ_k in Equation (2.51), we obtain $\sum_{i \in \mathcal{C}_k} E_i y_i \leq \tau_k \leq \sum_{i \in \mathcal{C}_k} L_i y_i, \forall k \in \mathcal{K}$, which are Constraints (2.29) in $LR(\mathcal{F}4)$.

Moreover, according to the definition of variable τ_k in Equation (2.51), Constraints (2.46) and (2.47) in $LR(\mathcal{F}4)$ are straightforwardly obtained from Constraints (2.41) and (2.42) in $LR(\mathcal{F}3)$ respectively.

Therefore, a feasible solution of $LR(\mathcal{F}3)$ is feasible for $LR(\mathcal{F}4)$.

2) Second, we demonstrate that a feasible solution of $LR(\mathcal{F}4)$ is feasible for $LR(\mathcal{F}3)$.

Let us consider a feasible solution of $LR(\mathcal{F}4)$. In order to show that this solution is feasible for $LR(\mathcal{F}3)$, we have to verify that it respects all the constraints of $LR(\mathcal{F}3)$. By comparing $LR(\mathcal{F}4)$ and $LR(\mathcal{F}3)$, we see that only Constraints (2.22), (2.41) and (2.42) in $LR(\mathcal{F}3)$ have to be verified.

Given a cluster $k \in \mathcal{K}$, Constraints (2.29) in $LR(\mathcal{F}4)$ is: $\sum_{i\in\mathcal{C}_k} E_i y_i \leq \tau_k \leq \sum_{i\in\mathcal{C}_k} L_i y_i$. Hence,

$$\exists \alpha_k \in [0,1], \tau_k = \alpha_k \sum_{i\in\mathcal{C}_k} E_i y_i + (1-\alpha_k) \sum_{i\in\mathcal{C}_k} L_i y_i. \tag{2.52}$$

Then, for $LR(\mathcal{F}3)$, we define the time variables t_i as follows:

$$t_i = \alpha_k E_i y_i + (1-\alpha_k) L_i y_i \qquad \forall k \in \mathcal{K}, i \in \mathcal{C}_k. \tag{2.53}$$

It is then clear that $E_i y_i \leq t_i \leq L_i y_i, \forall i \in \mathcal{V}$, which mean that Constraints (2.22) in $LR(\mathcal{F}3)$ are satisfied.

Moreover, according to the definition of time variables t_i in Equation (2.53), we have:

$$\sum_{i\in\mathcal{C}_k} t_i = \alpha_k \sum_{i\in\mathcal{C}_k} E_i y_i + (1-\alpha_k) \sum_{i\in\mathcal{C}_k} L_i y_i = \tau_k \qquad \forall k \in \mathcal{K}. \tag{2.54}$$

Then, by using Equation (2.54), Constraints (2.41) and (2.42) in $LR(\mathcal{F}3)$ are straightforwardly obtained from Constraints (2.46) and (2.47) in $LR(\mathcal{F}4)$ respectively.

Thus, a feasible solution of $LR(\mathcal{F}4)$ is feasible for $LR(\mathcal{F}3)$.

In conclusion, formulations $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$ are equivalent. □

To establish the dominance relation between $LR(\mathcal{F}2)$ and $LR(\mathcal{F}4)$, we need to prove the following Lemma first.

Lemma 2.4. *Given a feasible solution of $LR(\mathcal{F}2)$, the following constraints are valid:*

$$\sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k} x_{uv} + y_i + y_j - 2 \leq x_{ij} \qquad \forall (h,k) \in \mathcal{A}_c, i \in \mathcal{C}_h, j \in \mathcal{C}_k. \tag{2.55}$$

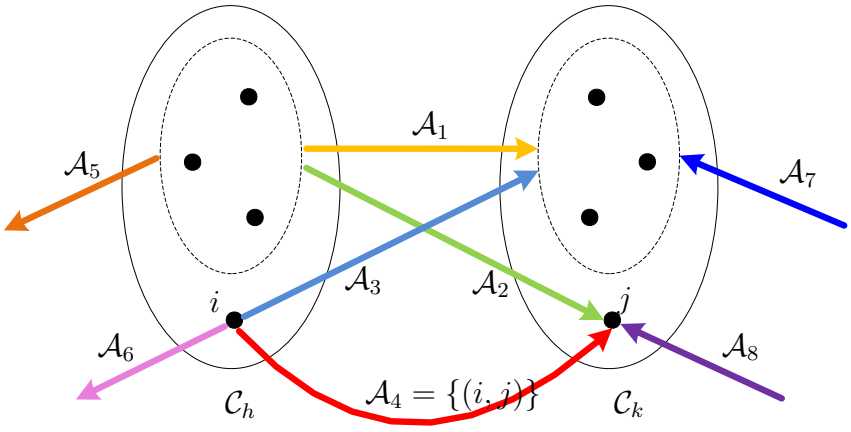


Figure 2.3: Illustration of the arc sets used in the proof of Lemma 2.4.

Proof. Let us consider two cluster h and k such that $(h,k) \in \mathcal{A}_c$, and two vertices i and j such that $i \in \mathcal{C}_h$, $j \in \mathcal{C}_k$. An illustration to ease the understating of the proof is provided in Figure 2.3. In this figure, the vertices in $\mathcal{C}_h \setminus \{i\}$ and $\mathcal{C}_k \setminus \{j\}$ are grouped in a dotted circle. In order to make the proof easy to understand, we divide the arcs leaving cluster $\mathcal{C}_h$ and the arcs entering cluster $\mathcal{C}_k$ into several sets defined as follows. $\mathcal{A}_1 = \{(u,v)|u \in \mathcal{C}_h \setminus \{i\}, v \in \mathcal{C}_k \setminus \{j\}\}$. $\mathcal{A}_2 = \{(u,j)|u \in \mathcal{C}_h \setminus \{i\}\}$. $\mathcal{A}_3 = \{(i,v)|v \in \mathcal{C}_k \setminus \{j\}\}$. $\mathcal{A}_4 = \{(i,j)\}$. $\mathcal{A}_5 = \{(u,v)|u \in \mathcal{C}_h \setminus \{i\}, v \in \mathcal{V} \setminus \mathcal{C}_k\}$. $\mathcal{A}_6 = \{(i,v)|v \in \mathcal{V}\setminus\mathcal{C}_k\}$. $\mathcal{A}_7 = \{(u,v)|u \in \mathcal{V}\setminus\mathcal{C}_h, v \in \mathcal{C}_k\setminus\{j\}\}$. $\mathcal{A}_8 = \{(u,j)|u \in \mathcal{V}\setminus\mathcal{C}_h\}$. Let $\widetilde{\mathcal{A}}$ be the set of all arcs going from $\mathcal{C}_h$ to $\mathcal{C}_k$, that is, $\widetilde{\mathcal{A}} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$.

Using Constraints (2.20)∼(2.21), and the aforementioned definition of the arc sets, we can write the left hand side of Constraints (2.55) as follows:

$$\begin{aligned}
&\sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k} x_{uv} + y_i + y_j - 2 \\
&= \sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k} x_{uv} + \sum_{(i,j')\in\delta^+(i)} x_{ij'} + \sum_{(i',j)\in\delta^-(j)} x_{i'j} - 2 \\
&= \sum_{(u,v)\in\widetilde{\mathcal{A}}} x_{uv} + \sum_{(u,v)\in\mathcal{A}_3\cup\mathcal{A}_4\cup\mathcal{A}_6} x_{uv} + \sum_{(u,v)\in\mathcal{A}_2\cup\mathcal{A}_4\cup\mathcal{A}_8} x_{uv} - 2 \\
&= \sum_{(u,v)\in\widetilde{\mathcal{A}}} x_{uv} + \sum_{(u,v)\in\mathcal{A}_2\cup\mathcal{A}_3\cup\mathcal{A}_4} x_{uv} + \sum_{(u,v)\in\mathcal{A}_4} x_{uv} + \sum_{(u,v)\in\mathcal{A}_6\cup\mathcal{A}_8} x_{uv} - 2
\end{aligned} \tag{2.56}$$

Then, using Constraints (2.19)∼(2.21), and the definition of the arc sets, the value 2 in Equation (2.56) can be replaced as follows:

$$\begin{aligned}
2 &= 1 + 1 \\
&= \sum_{i'\in\mathcal{C}_h} y_{i'} + \sum_{j'\in\mathcal{C}_k} y_{j'} \\
&= \sum_{i'\in\mathcal{C}_h} \sum_{(i',j')\in\delta^+(i')} x_{i'j'} + \sum_{j'\in\mathcal{C}_k} \sum_{(i',j')\in\delta^-(j')} x_{i'j'} \\
&= \sum_{(u,v)\in\widetilde{\mathcal{A}}\cup\mathcal{A}_5\cup\mathcal{A}_6} x_{uv} + \sum_{(u,v)\in\widetilde{\mathcal{A}}\cup\mathcal{A}_7\cup\mathcal{A}_8} x_{uv} \\
&= 2\sum_{(u,v)\in\widetilde{\mathcal{A}}} x_{uv} + \sum_{(u,v)\in\mathcal{A}_6\cup\mathcal{A}_8} x_{uv} + \sum_{(u,v)\in\mathcal{A}_5\cup\mathcal{A}_7} x_{uv}
\end{aligned} \tag{2.57}$$

Finally, using Equations (2.56) and (2.57), we have:

$$\begin{aligned}
\sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k} x_{uv} + y_i + y_j - 2 &= \sum_{(u,v)\in\mathcal{A}_2\cup\mathcal{A}_3\cup\mathcal{A}_4} x_{uv} + \sum_{(u,v)\in\mathcal{A}_4} x_{uv} - \sum_{(u,v)\in\widetilde{\mathcal{A}}} x_{uv} - \sum_{(u,v)\in\mathcal{A}_5\cup\mathcal{A}_7} x_{uv} \\
&= \sum_{(u,v)\in\mathcal{A}_4} x_{uv} - \sum_{(u,v)\in\mathcal{A}_1\cup\mathcal{A}_5\cup\mathcal{A}_7} x_{uv}
\end{aligned}$$

$$= x_{ij} - \sum_{(u,v)\in\mathcal{A}_1\cup\mathcal{A}_5\cup\mathcal{A}_7} x_{uv}$$
$$\leq x_{ij} \tag{2.58}$$

□

We can now prove the following dominance result.

Proposition 2.5. *Formulation $LR(\mathcal{F}2)$ is stronger than $LR(\mathcal{F}4)$.*

Proof. 1) First, we demonstrate that a feasible solution of $LR(\mathcal{F}2)$ is feasible for $LR(\mathcal{F}4)$.

Let us consider a feasible solution of $LR(\mathcal{F}2)$. In order to show that this solution is feasible for $LR(\mathcal{F}4)$, we need to verify that it respects all the constraints of $LR(\mathcal{F}4)$. By comparing $LR(\mathcal{F}2)$ and $LR(\mathcal{F}4)$, we see that only Constraints (2.35)∼(2.40), (2.46) and (2.47) in $\mathcal{F}4$ have to be verified.

For $LR(\mathcal{F}4)$, we define variables χ_{hk}, τ_{hk} and ς_{hk} as follows:

$$\chi_{hk} = \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} x_{ij} \qquad \forall (h,k)\in\mathcal{A}_c, \tag{2.59}$$
$$\tau_{hk} = \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} T_{ij}x_{ij} \qquad \forall (h,k)\in\mathcal{A}_c, \tag{2.60}$$
$$\varsigma_{hk} = \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} C_{ij}x_{ij} \qquad \forall (h,k)\in\mathcal{A}_c. \tag{2.61}$$

By summing up Constraints (2.20) of $\mathcal{F}2$: $\sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i$, $\forall i\in\mathcal{V}$, over all the vertices in $\mathcal{C}_h$, we have:

$$\text{left hand: } \sum_{i\in\mathcal{C}_h}\sum_{(i,j)\in\delta^+(i)} x_{ij} = \sum_{k\in\mathcal{K}\setminus\{h\}}\sum_{\substack{i\in\mathcal{C}_h\\ j\in\mathcal{C}_k}} x_{ij} = \sum_{k\in\mathcal{K}\setminus\{h\}} \chi_{hk} = \sum_{(h,k)\in\delta_c^+(h)} \chi_{hk} \tag{2.62}$$

$$\text{right hand: } \sum_{i\in\mathcal{C}_h} y_i = 1 \tag{2.63}$$

Thus, Constraints (2.35) of $LR(\mathcal{F}4)$ are satisfied. Similarly, we can obtain Constraints (2.36) of $LR(\mathcal{F}4)$ by summing up Constraints (2.21) of $LR(\mathcal{F}2)$ over all the vertices in $\mathcal{C}_k$.

Then, using Lemma 2.4, we demonstrate that Constraints (2.39) of $LR(\mathcal{F}4)$ are satisfied. We consider two clusters h and k such that $(h,k)\in\mathcal{A}_c$, and two vertices i and j such that $i\in\mathcal{C}_h, j\in\mathcal{C}_k$. By multiplying Constraints (2.55) by T_{ij}, we have:

$$\left(\sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k} x_{uv} + y_i + y_j - 2\right) T_{ij} \leq x_{ij}T_{ij}$$

$$\leq x_{ij}T_{ij} + \sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k, (u,v)\neq(i,j)} T_{uv}x_{uv}$$
$$\leq \sum_{u\in\mathcal{C}_h, v\in\mathcal{C}_k} T_{uv}x_{uv} \tag{2.64}$$

Using the variable definitions (2.59) and (2.60), Equation (2.64) leads to Constraints (2.39) of $LR(\mathcal{F}4)$: $(\chi_{hk} + y_i + y_j - 2)T_{ij} \leq \tau_{hk}$, $\forall (h,k) \in \mathcal{A}_c, i \in \mathcal{C}_h, j \in \mathcal{C}_k$.

Using variable definitions (2.59) and (2.60), it is easy to see that Constraints (2.40) of $LR(\mathcal{F}4)$ are satisfied.

The arguments used to prove that Constraints (2.39) and (2.40) are satisfied can be easily adapted by replacing T_{ij} by C_{ij}, and τ_{hk} by ς_{hk}. This permits to prove that Constraints (2.37) and (2.38) of $LR(\mathcal{F}4)$ are satisfied.

Using Constraints (2.30) of $LR(\mathcal{F}2)$, and the variable definitions (2.59) and (2.60), we have:

$$\text{left hand:} \quad \tau_h - \tau_k + \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} T_{ij}x_{ij} = \sum_{i\in\mathcal{C}_h} t_i - \sum_{j\in\mathcal{C}_k} t_j + \tau_{hk} \tag{2.65}$$

$$\begin{aligned}
\text{right hand:} & \sum_{i\in\mathcal{C}_h} L_i y_i - \sum_{j\in\mathcal{C}_k} E_j y_j - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} (L_i - E_j)x_{ij} \\
&= \sum_{i\in\mathcal{C}_h} \left(L_i \sum_{(i,j)\in\delta^+(i)} x_{ij} \right) - \sum_{j\in\mathcal{C}_k} \left(E_j \sum_{(i,j)\in\delta^-(j)} x_{ij} \right) - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} (L_i - E_j)x_{ij} \\
&= \sum_{i\in\mathcal{C}_h, j\notin\mathcal{C}_k} L_i x_{ij} + \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} L_i x_{ij} - \sum_{i\notin\mathcal{C}_h, j\in\mathcal{C}_k} E_j x_{ij} - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} E_j x_{ij} - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} (L_i - E_j)x_{ij} \\
&= \sum_{i\in\mathcal{C}_h, j\notin\mathcal{C}_k} L_i x_{ij} - \sum_{i\notin\mathcal{C}_h, j\in\mathcal{C}_k} E_j x_{ij} \\
&\leq L_h^{max} \sum_{i\in\mathcal{C}_h, j\notin\mathcal{C}_k} x_{ij} - E_k^{min} \sum_{i\notin\mathcal{C}_h, j\in\mathcal{C}_k} x_{ij} \\
&= L_h^{max} \left(1 - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} x_{ij} \right) - E_k^{min} \left(1 - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} x_{ij} \right) \\
&= (L_h^{max} - E_k^{min}) \left(1 - \sum_{i\in\mathcal{C}_h, j\in\mathcal{C}_k} x_{ij} \right) \\
&= (L_h^{max} - E_k^{min}) (1 - \chi_{hk}) \\
&= M_{hk}(1 - \tau_{hk})
\end{aligned} \tag{2.66}$$

This proves that Constraints (2.41) of $LR(\mathcal{F}4)$: $\sum_{i\in\mathcal{C}_h} t_i - \sum_{j\in\mathcal{C}_k} t_j + \tau_{hk} \leq M_{hk}(1-\chi_{hk})$ are satisfied.

Thus, all the Constraints of $LR(\mathcal{F}4)$ are satisfied, therefore a feasible solution of

$LR(\mathcal{F}2)$ is feasible for $LR(\mathcal{F}4)$.

2) Second, we show that a feasible solution of $LR(\mathcal{F}4)$ may not be feasible for $LR(\mathcal{F}2)$. Table 2.1 in Section 2.5.3 presents the gap in percentage of the linear relaxation of the four formulations, with respect to the optimal solution of the mixed integer formulations. For example, we consider the instance *TW_3burma14*. The optimal integer solution is 908. Formulation $LR(\mathcal{F}2)$ has a gap of 52.7%, i.e. the optimal value of $LR(\mathcal{F}2)$ is $908 \times (1 - 0.527) = 429.53$. Formulation $LR(\mathcal{F}4)$ has a gap of 59.3%, i.e. the optimal value of $LR(\mathcal{F}4)$ is $908 \times (1 - 0.593) = 369.11$. This proves that it does not exist a solution of $LR(\mathcal{F}2)$ with the same value of the optimal solution of $LR(\mathcal{F}4)$, since the optimal value of $LR(\mathcal{F}2)$ is greater than 369.11. Thus, from the optimal solution of $LR(\mathcal{F}4)$ an equivalent feasible solution for $LR(\mathcal{F}2)$ cannot be obtained.

In summary, every feasible solution of $LR(\mathcal{F}2)$ is feasible for $LR(\mathcal{F}4)$, but the opposite is not true. It means that $LR(\mathcal{F}2)$ is stronger than $LR(\mathcal{F}4)$. □

Note that, due to the equivalence between $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$, $LR(\mathcal{F}2)$ is stronger that $LR(\mathcal{F}3)$ as well.

Proposition 2.6. *There is no dominance relation between* $LR(\mathcal{F}1)$ *and* $LR(\mathcal{F}2)$.

Proof. 1) First, we show that a feasible solution of $LR(\mathcal{F}1)$ may not be feasible for $LR(\mathcal{F}2)$. Table 2.1 in Section 2.5.3 presents the gap in percentage of the linear relaxation of the four formulations, with respect to the optimal solution of the mixed integer formulations. For example, we consider the instance *TW_3burma14*. The optimal integer solution is 908. Formulation $LR(\mathcal{F}2)$ has a gap of 52.7%, i.e. the optimal value of $LR(\mathcal{F}2)$ is $908 \times (1 - 0.527) = 429.53$. Formulation $LR(\mathcal{F}1)$ has a gap of 53.1%, i.e. the optimal value of $LR(\mathcal{F}1)$ is $908 \times (1 - 0.531) = 426.23$. This proves that it does not exist a solution of $LR(\mathcal{F}2)$ with the same value of the optimal solution of $LR(\mathcal{F}1)$, since the optimal value of $LR(\mathcal{F}2)$ is greater than 426.23. Thus, from the optimal solution of $LR(\mathcal{F}1)$ an equivalent feasible solution for $LR(\mathcal{F}2)$ cannot be obtained.

2) Second, we show that a feasible solution of $LR(\mathcal{F}2)$ may not be feasible for $LR(\mathcal{F}1)$. From Table 2.1, we consider for example the instance *TW_5gr24*. The optimal integer solution is 263. Formulation $LR(\mathcal{F}1)$ has a gap of 15.3%, i.e. the optimal value of $LR(\mathcal{F}1)$ is $263 \times (1 - 0.153) = 222.70$. Formulation $LR(\mathcal{F}2)$ has a gap of 22.0%, i.e. the optimal value of $LR(\mathcal{F}2)$ is $263 \times (1 - 0.22) = 205.24$. This proves that it does not exist a solution of $LR(\mathcal{F}1)$ with the same value of the optimal solution of $LR(\mathcal{F}2)$, since the optimal value of $LR(\mathcal{F}1)$ is greater than 205.24. Thus, from the optimal solution of $LR(\mathcal{F}2)$ an equivalent feasible solution for $LR(\mathcal{F}1)$ cannot be obtained. □

Note that, by comparing formulations $LR(\mathcal{F}1)$ and $LR(\mathcal{F}2)$, Constraints (2.23) in $LR(\mathcal{F}1)$ cannot be straightforwardly transformed into Constraints (2.30) in $LR(\mathcal{F}2)$. Constraints (2.23) in $LR(\mathcal{F}1)$ are: $t_i - t_j + T_{ij}x_{ij} \leq L_i y_i - E_j y_j - (L_i - E_j)x_{ij}, \forall (i,j) \in \mathcal{A}, j \neq 0$. Let us consider $i \in \mathcal{C}_h$, $j \in \mathcal{C}_k$. By summing up Constraints (2.23) over all the arcs between $\mathcal{C}_h$ and $\mathcal{C}_k$, we obtain:

$$
\begin{aligned}
&|\mathcal{C}_k| \sum_{i \in \mathcal{C}_h} t_i - |\mathcal{C}_h| \sum_{j \in \mathcal{C}_k} t_j + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} T_{ij} x_{ij} \\
&= |\mathcal{C}_k| \tau_h - |\mathcal{C}_h| \tau_k + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} T_{ij} x_{ij} \\
&\leq |\mathcal{C}_k| \sum_{i \in \mathcal{C}_h} L_i y_i - |\mathcal{C}_h| \sum_{j \in \mathcal{C}_k} E_j y_j - \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} (L_i - E_j) x_{ij} \\
&\qquad\qquad \forall h \in \mathcal{K}, k \in \mathcal{K} \setminus \{0\}, h \neq k. \qquad (2.67)
\end{aligned}
$$

These inequalities are different from Constraints (2.30) in $LR(\mathcal{F}2)$.

2.5 Experimental results

In this section, we conduct computational experiments to compare the four formulations, and to assess the quality of the supervalid inequalities (2.28) and (2.33) in formulations $\mathcal{F}1$ and $\mathcal{F}2$ respectively. We report two types of computational results: the comparison of the LP relaxations and the performance of a branch-and-bound algorithm to solve the MIPs. To solve the LPs and MIPs, we use CPLEX 12.6.3. on a PC Intel(R) Core(TM) i5-6200U CPU 2.20GHz and 64G RAM.

2.5.1 Testbed

We use the instances created by Yuan *et al.* (2019a) (detailed in Chapter 3.6.1) for the GTSPTW. The first group of instances, denoted as *G1*, is created by performing suitable modifications to the existing benchmark for the generalized traveling salesman problem (Karapetyan, 2012). The second group of instances, denoted as *G2*, is generated as *G1*, except the maximum number of vertices per cluster is fixed to 5. From these sets of instances, we exclude instance that were not solved to optimality by the branch-and-cut developed by Yuan *et al.* (2019a). We also exclude two instances where the MTZ constraints cannot eliminate some subtours (see the last paragraph of Section 2.3). Hence, we consider 39 instances in set *G1*, and 33 instances in set *G2*.

To test the branch-and-bound performances of the MIP formulations, we only conduct experiments on the instances of *G1* and *G2* with no more than 20 clusters.

2.5.2 Preprocessing

Before solving the problem, we perform a preprocessing step. It consists in tightening the TWs and eliminating vertices and arcs that cannot be part of a feasible solution.

The TW width can be reduced by taking into account the earliest and the latest arrival and departure times at each vertex of the graph from or to another vertex. In particular, we consider the following conditions proposed by Desrochers *et al.* (1992):

- earliest arrival time from predecessors: $E_i = \max\left\{E_i, \min\left\{L_i, \min_{(j,i)\in\mathcal{A}}\left\{E_j + T_{ji}\right\}\right\}\right\}$;
- earliest departure time to successors: $E_i = \max\left\{E_i, \min\left\{L_i, \min_{(i,j)\in\mathcal{A}}\left\{E_j - T_{ij}\right\}\right\}\right\}$;
- latest arrival time from predecessors: $L_i = \min\left\{L_i, \max\left\{E_i, \max_{(j,i)\in\mathcal{A}}\left\{L_i + T_{ji}\right\}\right\}\right\}$;
- latest departure time to successors: $L_i = \min\left\{L_i, \max\left\{E_i, \max_{(i,j)\in\mathcal{A}}\left\{L_j - T_{ij}\right\}\right\}\right\}$.

These conditions are applied iteratively to all vertices until no TW can be reduced. After applying the TW reduction, we sparsify the graph by eliminating the vertices and arcs that cannot be part of a feasible solution. Let SP_{ij} denote the shortest traveling time between vertices i and j. When the triangle inequality is not satisfied for traveling times, the shortest traveling time SP_{ij} going from vertex i to vertex j can include the visit of other vertices and can be lower than T_{ij}. We remove:

- a vertex $i \in \mathcal{V}$ if a round trip from the depot to the vertex leads to a time window violation, i.e. $E_0 + T_{0i} > L_i$ or $\max\left\{E_0 + T_{0i}, E_i\right\} + T_{i0} > L_0$;
- an arc $(i,j) \in \mathcal{A}$ if $E_i + T_{ij} > L_j$ or if the route that starts from the depot, visits vertex i, then vertex j then goes back to the depot violates at least one TW, i.e., $\max\left\{E_0 + SP_{0i}, E_i\right\} + T_{ij} > L_j$ or $\max\left\{\max\left\{E_0 + SP_{0i}, E_i\right\} + T_{ij}, E_j\right\} + SP_{j0} > L_0$.

2.5.3 Comparison of the LP relaxations

In this section, we discuss the results obtained by solving the LP relaxations of the four formulations. The results of instances in sets *G1* and *G2* are reported in Tables 2.1 and 2.2 respectively.

First, we describe the column headings used in Tables 2.1 and 2.2. The first column *Instance* indicates the instance name. The number in the middle of the name represents the number of clusters, and the number at the end represents the number of vertices, both excluding the depot. The second column *OptValue* indicates the optimal value of an integer solution for the corresponding instance. The next columns show the optimality gap in percentage between the optimal value and the lower bound obtained by solving an LP relaxation, which is calculated as $(OptValue - lower\,bound)/OptValue * 100$. These gaps are presented for the LP relaxations of the four formulations in columns $LR(\mathcal{F}1)$, $LR(\mathcal{F}2)$, $LR(\mathcal{F}3)$, and $LR(\mathcal{F}4)$. In the last two columns $LR(\mathcal{F}1) + SVI$ and $LR(\mathcal{F}2) + SVI$, we report the optimality gaps when solving $LR(\mathcal{F}1)$ with supervalid inequalities (2.28), and $LR(\mathcal{F}2)$ with supervalid inequalities (2.33) respectively. In each row, we mark in bold the best result obtained from the LP relaxations of the four basic formulations. Note that the strongest formulation, with the highest lower bound, is the one with the smallest optimality gap. In each row, we also mark in bold the best result obtained from all the LP relaxations including the ones with supervalid inequalities. In the last row of the tables, we indicate the average results over all the instances. Note that we do not report computation times since the LP relaxations are fast to solve. The computation times are always less than 3 seconds.

From Tables 2.1 and 2.2, it can be seen that the lower bounds obtained from $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$ are always the same. The lower bounds obtained from $LR(\mathcal{F}2)$ are always better than those obtained from $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$ since the optimality gaps are lower. For some instances, the lower bounds obtained from $LR(\mathcal{F}1)$ are better than those obtained from $LR(\mathcal{F}2)$, e.g., TW_5gr21 in *G1* and TW_7gr24 in *G2*, while for some instances, the reverse is true, e.g., TW_3burma14 in *G1* and TW_8ulysses22 in *G2*. The above observations are in accordance with the theoretical results presented in Section 2.4. From the average results in the last row of Table 2.1 and 2.2, we can see that in average $LR(\mathcal{F}1)$ and $LR(\mathcal{F}2)$ are stronger than $LR(\mathcal{F}3)$ and $LR(\mathcal{F}4)$. On average, $LR(\mathcal{F}1)$ is slightly stronger than $LR(\mathcal{F}2)$. The best lower bounds are obtained with $LR(\mathcal{F}1)$ for 62 instances, and they are obtained with $LR(\mathcal{F}2)$ for 15 instances.

Table 2.1: Optimality gaps obtained for LP relaxations on instances in *G1*.

Instance	OptValue	GAP(%)					
		$LR(\mathcal{F}1)$	$LR(\mathcal{F}2)$	$LR(\mathcal{F}3)$	$LR(\mathcal{F}4)$	$LR(\mathcal{F}1)+SVI$	$LR(\mathcal{F}2)+SVI$
TW_3burma14	908	53.06	**52.69**	59.35	59.35	**42.24**	44.44
TW_4gr17	962	**34.99**	35.08	46.78	46.78	**33.26**	34.69
TW_4ulysses16	2392	**59.41**	**59.41**	61.57	61.57	59.41	59.41
TW_5gr21	1165	**32.02**	34.12	47.67	47.67	**27.60**	32.96
TW_5gr24	263	**15.32**	21.96	34.24	34.24	**8.06**	16.48
TW_5ulysses22	3287	35.25	**34.69**	35.08	35.08	31.11	**30.32**
TW_6bayg29	476	**31.86**	31.95	40.56	40.56	**29.69**	30.26
TW_6bays29	628	**37.77**	38.30	44.02	44.02	**32.07**	35.98
TW_6fri26	354	**28.45**	30.50	31.56	31.56	**20.83**	28.78
TW_7ftv33	416	**26.91**	27.42	41.09	41.09	**18.35**	21.92
TW_8ftv36	538	44.18	**43.75**	56.14	56.14	36.42	**31.72**
TW_8ftv38	384	**29.35**	30.59	50.62	50.62	**26.34**	28.24
TW_9dantzig42	322	**37.43**	37.94	44.74	44.74	32.80	**32.54**
TW_10att48	4113	**41.60**	42.65	45.80	45.80	**33.67**	37.90
TW_10gr48	1437	**41.95**	42.50	43.51	43.51	**34.85**	39.62
TW_10hk48	5268	**43.06**	43.46	49.62	49.62	**36.94**	39.20
TW_11berlin52	3632	**35.36**	36.07	42.40	42.40	**32.40**	32.71
TW_11eil51	151	**40.51**	41.55	47.57	47.57	**33.51**	40.21
TW_12brazil58	13503	51.54	**51.08**	52.92	52.92	48.50	**40.59**
TW_14st70	289	**50.54**	50.73	53.91	53.91	**44.53**	48.75
TW_16eil76	205	**39.19**	39.23	50.64	50.64	**35.92**	37.98
TW_16pr76	57164	**38.98**	39.08	43.12	43.12	33.27	**29.41**
TW_20gr96	33128	**44.72**	45.50	51.87	51.87	**38.58**	42.85
TW_20kroA100	10209	**41.37**	41.69	54.09	54.09	**39.48**	40.81
TW_20kroB100	9862	**45.03**	45.33	50.73	50.73	**32.82**	35.06
TW_20kroC100	9728	**43.81**	44.16	51.52	51.52	**38.12**	39.78
TW_20kroD100	9210	**53.94**	54.23	60.37	60.37	**45.45**	**45.42**
TW_20kroE100	9514	**36.42**	37.29	43.64	43.64	**30.29**	35.68
TW_20rat99	478	**49.06**	49.41	54.95	54.95	**40.81**	45.91
TW_20rd100	3446	**41.75**	41.78	44.71	44.71	**34.64**	34.70
TW_21eil101	247	40.02	**39.97**	44.45	44.45	**36.48**	36.70
TW_21lin105	7582	37.97	**37.92**	38.62	38.62	32.59	**30.25**
TW_22pr107	21352	**52.23**	52.49	53.62	53.62	**46.81**	49.46
TW_24gr120	2811	51.60	**51.41**	55.60	55.60	**46.55**	47.45
TW_25pr124	36520	**56.11**	56.20	63.44	63.44	49.38	**46.62**
TW_26ch130	2891	**38.55**	38.90	44.65	44.65	**33.30**	34.89
TW_28gr137	34123	**47.06**	47.31	52.46	52.46	**34.94**	35.73
TW_29pr144	43639	29.35	**29.34**	31.79	31.79	26.13	**25.90**
TW_30kroA150	11475	**45.66**	45.90	51.71	51.71	**40.83**	41.70
Average		41.11	41.63	47.98	47.98	35.36	37.00

Table 2.2: Optimality gaps obtained for LP relaxations on instances in *G2*.

Instance	OptValue	GAP(%)					
		$LR(\mathcal{F}1)$	$LR(\mathcal{F}2)$	$LR(\mathcal{F}3)$	$LR(\mathcal{F}4)$	$LR(\mathcal{F}1)+SVI$	$LR(\mathcal{F}2)+SVI$
TW_4burma14	957	**26.75**	**26.75**	45.62	45.62	**23.99**	26.56
TW_5gr17	766	**19.23**	**19.23**	20.10	20.10	18.47	**18.37**
TW_6ulysses16	2521	**53.59**	**53.59**	63.66	63.66	**53.53**	**53.53**
TW_7gr21	1475	**26.68**	27.56	33.42	33.42	**22.90**	26.59
TW_7gr24	365	**24.07**	25.32	36.55	36.55	**16.83**	21.72
TW_8ulysses22	3461	43.96	**43.08**	50.58	50.58	43.12	**36.08**
TW_8bayg29	515	**25.20**	26.64	35.37	35.37	**17.69**	22.12
TW_8bays29	595	**18.90**	19.34	22.29	22.29	**15.29**	16.99
TW_9fri26	498	**30.98**	31.61	36.34	36.34	22.80	**21.85**
TW_10ftv33	467	**22.68**	23.46	42.15	42.15	**18.70**	19.25
TW_10ftv36	500	**31.67**	32.75	47.25	47.25	27.68	**27.36**
TW_10ftv38	469	**37.49**	38.35	50.26	50.26	**34.46**	35.51
TW_12dantzig42	364	29.22	**29.14**	37.18	37.18	26.62	**25.37**
TW_13gr48	2010	**39.12**	39.63	47.31	47.31	**36.54**	37.75
TW_14att48	5415	**46.38**	47.98	49.65	49.65	**39.00**	44.68
TW_14hk48	5989	**42.54**	42.73	48.27	48.27	**37.19**	39.51
TW_15eil51	172	**33.86**	34.14	43.21	43.21	**25.16**	29.03
TW_16berlin52	3821	**30.86**	30.94	36.34	36.34	28.60	**28.40**
TW_17brazil58	12665	**42.16**	42.19	47.53	47.53	**33.27**	34.11
TW_20st70	329	**40.01**	40.23	45.21	45.21	**33.54**	34.65
TW_21eil76	245	**46.80**	47.19	51.80	51.80	**40.50**	44.33
TW_22pr76	71098	**40.16**	40.33	47.71	47.71	33.84	**31.55**
TW_24kroC100	9932	**50.39**	50.49	53.44	53.44	**41.11**	42.80
TW_26kroA100	10515	**51.57**	51.79	55.22	55.22	**38.92**	42.65
TW_28rat99	618	**47.20**	47.44	52.03	52.03	**38.65**	41.84
TW_28kroB100	11937	**44.84**	45.19	50.47	50.47	**35.43**	39.00
TW_28kroD100	10720	**54.69**	55.17	57.59	57.59	47.90	**46.41**
TW_29gr96	35064	**45.78**	**45.78**	47.41	47.41	40.43	**39.86**
TW_29kroE100	11275	**42.31**	42.54	47.64	47.64	**33.12**	35.62
TW_29rd100	4380	**40.88**	40.99	46.91	46.91	**33.46**	34.88
TW_30eil101	292	**35.74**	35.84	40.72	40.72	**31.09**	32.54
TW_36ch130	3309	**35.93**	36.03	39.84	39.84	30.64	**29.83**
TW_38ch150	3175	**39.68**	40.14	48.43	48.43	**32.01**	35.48
Average		37.62	37.99	44.77	44.77	31.89	33.22

In Tables 2.1 and 2.2, we also report the results when solving $LR(\mathcal{F}1)$ and $LR(\mathcal{F}2)$ with supervalid inequalities (2.28) and (2.33) respectively. We observe that using the supervalid inequalities clearly improve the quality of the lower bounds. The average optimality gaps obtained from $LR(\mathcal{F}1)$ with supervalid inequalities are a bit better than those obtained from $LR(\mathcal{F}2)$ with supervalid inequalities.

2.5.4 Branch-and-bound performance

The experimental comparison of lower bounds indicates that formulations $\mathcal{F}1$ and $\mathcal{F}2$ are stronger than formulations $\mathcal{F}3$ and $\mathcal{F}4$ for the GTSPTW. In this section, we conduct additional experiments to compare the efficiency of the four MIP formulations and the use of supervalid inequalities when using them to solve the problem with CPLEX's branch-and-bound scheme. The tests are performed on instances with no more than 20 clusters. CPLEX is used with default parameters, and we set a computational time limit (TL) of 30 minutes.

The results of instances in *G1* and *G2* are shown in Tables 2.3 and 2.4 respectively. The column headings are the followings. The first two columns, as in Tables 2.1 and 2.2, are the instance name and the optimal value. Then, we report the optimality gap in percentage and the computation time in seconds for each formulation. If the computation time is lower than the time limit, the optimality gap is 0. If the time limit has been reached, this is indicated by *TL*, and the optimality gap is calculated as $(OptValue - lower\,bound)/OptValue * 100$. We first provide results for formulations $\mathcal{F}1$, $\mathcal{F}2$, $\mathcal{F}3$ and $\mathcal{F}4$. Then we provide results for formulations $\mathcal{F}1$ with supervalid inequalities (2.28), and $\mathcal{F}2$ with supervalid inequalities (2.33) respectively. When the time limit has been reached for an instance, we mark in bold the best optimality gap obtained by the four basic formulations. We also mark in bold the best optimality gap obtained by all the formulations including the ones with supervalid inequalities. In the last row of the tables, we indicate the average results over all the instances.

From Tables 2.3 and 2.4, by comparing the results obtained with $\mathcal{F}3$ and $\mathcal{F}4$, we can see that they can solve the same instances to optimality, both solving 32 out of 46 instances in total. For instances that are not solved to optimality with formulations $\mathcal{F}3$ and $\mathcal{F}4$, formulation $\mathcal{F}4$ can always provide a slightly better optimality gap. From the average results in the last row, formulation $\mathcal{F}4$ provides, on average, a smaller optimality gap than formulation $\mathcal{F}3$, and requires shorter computation times.

Table 2.3: Optimality gaps and computation times to solve MIP formulations for instances in *G1*.

Instance	OptValue	$\mathcal{F}1$		$\mathcal{F}2$		$\mathcal{F}3$		$\mathcal{F}4$		$\mathcal{F}1 + SVI$		$\mathcal{F}2 + SVI$	
		GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)
TW_3burma14	908	0	0.15	0	0.14	0	0.17	0	0.17	0	0.15	0	0.11
TW_4gr17	962	0	0.29	0	0.21	0	0.23	0	0.19	0	0.32	0	0.22
TW_4ulysses16	2392	0	0.22	0	0.16	0	0.25	0	0.19	0	0.26	0	0.22
TW_5gr21	1165	0	0.27	0	0.17	0	0.23	0	0.27	0	0.18	0	0.20
TW_5gr24	263	0	0.17	0	0.19	0	0.40	0	0.26	0	0.19	0	0.17
TW_5ulysses22	3287	0	0.32	0	0.46	0	0.39	0	0.28	0	0.43	0	0.36
TW_6bayg29	476	0	0.54	0	0.42	0	0.69	0	0.63	0	0.54	0	0.35
TW_6bays29	628	0	0.52	0	0.39	0	1.50	0	1.21	0	0.68	0	0.42
TW_6fri26	354	0	0.41	0	0.30	0	0.55	0	0.41	0	0.57	0	0.43
TW_7ftv33	416	0	0.73	0	0.45	0	3.05	0	2.49	0	0.75	0	0.35
TW_8ftv36	538	0	1.29	0	0.71	0	7.73	0	7.13	0	1.35	0	0.60
TW_8ftv38	384	0	0.82	0	0.47	0	4.46	0	3.88	0	0.85	0	0.43
TW_9dantzig42	322	0	6.73	0	4.30	0	34.17	0	22.80	0	3.47	0	2.07
TW_10att48	4113	0	12.93	0	9.83	0	37.62	0	27.99	0	8.07	0	2.65
TW_10gr48	1437	0	16.88	0	27.73	0	189.43	0	229.43	0	7.18	0	2.38
TW_10hk48	5268	0	19.99	0	8.77	0	299.94	0	201.17	0	9.63	0	7.16
TW_11berlin52	3632	0	81.65	0	38.70	0	153.95	0	145.75	0	74.07	0	31.54
TW_11eil51	151	0	9.77	0	11.44	0	1025.92	0	862.97	0	8.26	0	6.09
TW_12brazil58	13503	0	246.18	0	212.55	0	66.07	0	56.73	0	95.29	0	110.37
TW_14st70	289	30.69	TL	**19.90**	TL	44.01	TL	40.93	TL	19.61	TL	**8.21**	TL
TW_16eil76	205	19.28	TL	**0**	1044.80	40.20	TL	33.74	TL	0	437.33	0	631.78
TW_16pr76	57164	19.50	TL	**15.11**	TL	26.10	TL	25.52	TL	**0**	691.50	**0**	185.22
TW_20gr96	33128	27.30	TL	**26.05**	TL	43.46	TL	38.61	TL	**20.95**	TL	21.11	TL
TW_20kroA100	10209	28.27	TL	**25.34**	TL	47.62	TL	45.86	TL	**23.28**	TL	24.19	TL
TW_20kroB100	9862	**25.57**	TL	26.47	TL	43.24	TL	39.27	TL	17.97	TL	**17.79**	TL
TW_20kroC100	9728	**28.94**	TL	31.46	TL	41.99	TL	39.89	TL	17.42	TL	**16.64**	TL
Average		6.91	500.00	5.55	467.78	11.02	554.88	10.15	544.77	3.82	397.73	3.38	383.97

By comparing results of formulations $\mathcal{F}1$ and $\mathcal{F}2$, we can see that formulation $\mathcal{F}2$ can solve to optimality one instance more in set *G1* (TW_16eil76). From the average results in the last row, formulation $\mathcal{F}2$ is able to obtain, on average, smaller optimality gap than formulation $\mathcal{F}1$, and requires shorter computation times.

From Tables 2.3 and 2.4, we can see that formulations $\mathcal{F}1$ and $\mathcal{F}2$ can solve more instances to optimality than formulations $\mathcal{F}3$ and $\mathcal{F}4$. The optimality gaps obtained with formulations $\mathcal{F}1$ and $\mathcal{F}2$ are always better than those obtained with formulations $\mathcal{F}3$ and $\mathcal{F}4$. The average computation times with formulations $\mathcal{F}1$ and $\mathcal{F}2$ are really shorter than the ones for formulations $\mathcal{F}3$ and $\mathcal{F}4$. In conclusion, with regard to the branch-and-bound performance to solve the GTSPTW, formulation $\mathcal{F}2$ is a bit better than formulation $\mathcal{F}1$, and formulations $\mathcal{F}1$ and $\mathcal{F}2$ are clearly better than formulations $\mathcal{F}3$ and $\mathcal{F}4$.

In Tables 2.3 and 2.4, we also report the results when solving the problem using $\mathcal{F}1$ and $\mathcal{F}2$ with supervalid inequalities (2.28) and (2.33) respectively. Compared with the

results for $\mathcal{F}1$ and $\mathcal{F}2$, it can be seen that $\mathcal{F}1$ and $\mathcal{F}2$ with supervalid inequalities can respectively solve two (TW_16eil76, TW_16pr76) and one (TW_16pr76) instances more to optimality. Using valid inequalities reduces the optimality gaps for instances not solved to optimality with formulations $\mathcal{F}1$ and $\mathcal{F}2$. The average optimality gaps are improved for both $\mathcal{F}1$ and $\mathcal{F}2$, and the average computation times strongly decrease. It can be concluded that the proposed supervalid inequalities strengthen formulations $\mathcal{F}1$ and $\mathcal{F}2$.

Table 2.4: Optimality gaps and computation times to solve MIP formulations for instances in *G2*.

		$\mathcal{F}1$		$\mathcal{F}2$		$\mathcal{F}3$		$\mathcal{F}4$		$\mathcal{F}1 + SVI$		$\mathcal{F}2 + SVI$	
Instance	OptValue	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)	GAP (%)	time (s)
TW_4burma14	957	0	0.11	0	0.09	0	0.25	0	0.16	0	0.13	0	0.12
TW_5gr17	766	0	0.18	0	0.19	0	0.26	0	0.24	0	0.17	0	0.13
TW_6ulysses16	2521	0	0.19	0	0.28	0	0.26	0	0.20	0	0.22	0	0.26
TW_7gr21	1475	0	0.44	0	0.24	0	0.52	0	0.40	0	0.36	0	0.30
TW_7gr24	365	0	0.45	0	0.50	0	0.56	0	0.49	0	0.38	0	0.34
TW_8ulysses22	3461	0	0.52	0	0.41	0	0.60	0	0.63	0	0.45	0	0.44
TW_8bayg29	515	0	0.54	0	0.45	0	1.91	0	1.10	0	0.57	0	0.45
TW_8bays29	595	0	0.63	0	0.58	0	0.86	0	1.45	0	0.61	0	0.47
TW_9fri26	498	0	0.91	0	0.32	0	0.99	0	0.80	0	0.78	0	0.47
TW_10ftv33	467	0	1.36	0	0.64	0	9.34	0	5.63	0	1.52	0	1.10
TW_10ftv36	500	0	1.41	0	1.04	0	16.47	0	13.10	0	2.23	0	0.90
TW_10ftv38	469	0	2.36	0	1.20	0	84.20	0	62.81	0	1.59	0	0.88
TW_12dantzig42	364	0	8.34	0	7.30	0	174.74	0	123.12	0	4.36	0	3.81
TW_13gr48	2010	0	68.03	0	37.61	13.86	TL	11.96	TL	0	27.74	0	33.46
TW_14att48	5415	0	1404.77	0	643.87	27.40	TL	22.19	TL	0	913.73	0	365.66
TW_14hk48	5989	0	841.26	0	407.86	24.95	TL	23.47	TL	0	363.93	0	111.68
TW_15eil51	172	0	244.00	0	54.94	15.65	TL	14.63	TL	0	12.09	0	6.80
TW_16berlin52	3821	0	532.38	0	192.36	8.63	TL	5.23	TL	0	139.32	0	66.87
TW_17brazil58	12665	0	1022.76	0	710.93	20.78	TL	20.19	TL	0	257.60	0	133.90
TW_20st70	329	23.23	TL	**19.93**	TL	31.66	TL	29.44	TL	**12.20**	TL	15.33	TL
Average		1.16	296.53	1.00	193.04	7.15	644.55	6.36	640.51	0.61	176.39	0.77	126.40

2.6 Conclusions

In this paper, we present four integer linear programming formulations for the Generalized Traveling Salesman Problem with Time Windows (GTSPTW) arising in the last mile delivery context. The models differ in the way they define the arc variables and time variables: based on vertices or clusters. Dominance relations between the LP relaxations of these formulations are established theoretically. Computational results on LP relaxations show that on average formulation $\mathcal{F}1$ is the best, followed by formulation $\mathcal{F}2$. However, when solving the GTSPTW with the branch-and-bound scheme

implemented in CPLEX, on average formulation $\mathcal{F}2$ is the most efficient, followed by formulation $\mathcal{F}1$. Therefore, we recommend using formulations $\mathcal{F}1$ and $\mathcal{F}2$ for the solution of the GTSPTW. Moreover, supervalid inequalities for formulations $\mathcal{F}1$ and $\mathcal{F}2$ are proposed and can strengthen them significantly.

Chapter 3

A branch-and-cut algorithm for the GTSPTW

Contents

This chapter corresponds to the paper "A branch-and-cut algorithm for the generalized traveling salesman problem with time windows", submitted to European Journal of Operational Research on 4 April, 2019 and received major revision on 10 August, 2019.

Abstract: The generalized traveling salesman problem with time windows (GTSPTW) is defined on a directed graph where the vertex set is partitioned into clusters. One cluster contains only the depot. Each vertex is associated with a time interval, the time window, during which the visit must take place if the vertex is visited. The objective is to find a minimum cost tour starting and ending at the depot such that each cluster is visited exactly once and time constraints are respected, i.e., for each cluster, one vertex is visited during its time window. In this paper, two integer linear programming formulations for GTSPTW are provided as well as several problem-specific valid inequalities. A branch-and-cut algorithm is developed in which the inequalities are separated dynamically. To reduce the computation times, an initial upper bound is provided by a simple and fast heuristic. We propose different sets of instances characterized by their time window structures. Experimental results show that our algorithm is effective and instances including up to 30 clusters can be solved to optimality within one hour.

3.1 Introduction

E-commerce is a thriving market around the world and is very well suited to the busy lifestyle of today's customers. An annual survey conducted by the analytics firm comScore and UPS revealed that American consumers purchased more things online than in stores in 2016 (Farber, 2016). eMarketer estimated that e-commerce sales would reach $4 trillion in 2020 (eMarketer, 2018). It is clear that this growing e-commerce poses a major challenge to transportation companies, especially with regard to last mile delivery. Nowadays, the most common delivery service is home/workplace

delivery (Lowe & Rigby, 2014). Customers wait at home or at work to get their online orders. Besides, companies like Amazon and FedEx are developing locker delivery. When customers shop online, they can choose a nearby locker as their pick-up location. In the past two years, a new concept called *trunk delivery* has been introduced. Here, customers' orders can be delivered to the trunks of their cars. Volvo launched its world-first in-car delivery service in Sweden in 2016. The courier has a one-time digital code to get access to the trunk of the car. Trunk delivery is different from home delivery and locker delivery since the car moves during the day and may be in different locations during different periods of time. As a consequence, synchronization between the car and the courier is required to perform the delivery. In this article, we provide two mathematical programming models, and we develop an efficient exact method for the last mile delivery problem that combines all these delivery services: home/workplace, locker, and car trunk. We focus on the one vehicle case, i.e., we assume that a single vehicle can deliver all the customers on the same route. In Figure 3.1, we give an example of the real-life case. Four customers are represented with their associated locations into a dotted circle. Every possible delivery location has a time interval that represents the time window (TW). In the case of a home or trunk delivery, the TW represents when the customer or his/her car would be present at that location. In the case of a locker, the TW represents the period the courier can deliver the parcel before the customer picks it up. The problem consists in determining jointly the location visited for each customer and the sequence of visits while satisfying the TW restrictions.

The problem addressed in this paper is the generalized traveling salesman problem with time windows (GTSPTW). To the best of our knowledge, this problem has not been studied yet. It is related to the generalized traveling salesman problem (GTSP) (Fischetti *et al.*, 1997) where TWs are not present, and to the traveling salesman problem with time windows (TSPTW) where all clusters contain a single vertex. It is also related to the generalized vehicle routing problem with time windows (GVRPTW), which is the multi-vehicle case of the problem studied in this paper.

This article aims to provide an efficient exact solution method for the GTSPTW. The main contributions of the paper are as follows: 1) we study a new problem and present two formulations for the GTSPTW. This problem is of great interest in the context of last mile delivery, 2) we propose several valid inequalities for GTSPTW, 3) we develop procedures to separate these inequalities within a branch-and-cut algorithm,

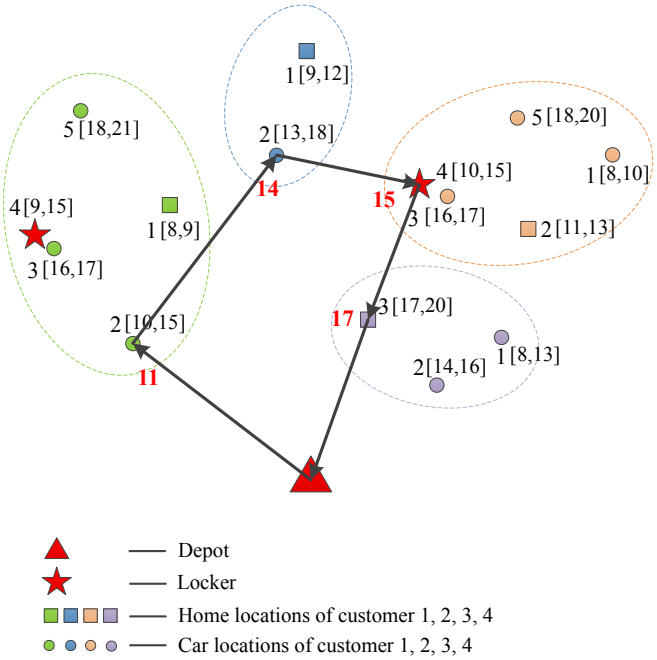


Figure 3.1: An example of GTSPTW instance.

4) we present a simple and fast heuristic for GTSPTW to get an initial solution, 5) we assess the efficiency of our algorithm on different sets of instances that we generated for the GTSPTW.

The remainder of this paper is organized as follows. A formal description of the problem and two mathematical models are provided in Section 3.3. Section 3.2 presents the related literature. Section 3.4 describes some valid inequalities for the GTSPTW. A general framework of the branch-and-cut algorithm is given in Section 3.5, including preprocessing, an initial heuristic to compute an upper bound, and separation procedures for the proposed valid inequalities. Section 3.6 gives details about the generation of three groups of instances and reports the computational results. Finally, conclusions are drawn in Section 3.7.

3.2 Literature review

To the best of our knowledge, there is no existing literature on the GTSPTW. However, there exist works addressing the GTSP. The GTSP introduced by Srivastava *et al.* (1969) is defined on a graph where the vertex set is partitioned into clusters. The problem consists in finding a minimum cost tour which visits exactly one vertex of

each cluster. In the GTSP, TWs are not present. In the literature, there are various approaches to solve the GTSP.

One approach is to transform an instance of the GTSP into an instance of the well-studied Traveling Salesman Problem (TSP), and then solve it by applying algorithms for the TSP (Dimitrijević & Šarić, 1997; Laporte & Semet, 1999; Noon & Bean, 1993). At first glance, this approach seems promising. However, the resulting instances are difficult to solve for the existing TSP solvers since the produced instances have a rather unusual structure, and a near-optimal solution for the TSP instance may correspond to an infeasible solution for the related GTSP instance (Karapetyan & Gutin, 2012).

Another approach consists in developing exact algorithms. However, the existing literature is quite limited. Srivastava *et al.* (1969) proposed a dynamic programming approach. Noon & Bean (1991) presented a branch-and-bound approach for the asymmetric GTSP (AGTSP). They proposed a Lagrangian relaxation to compute a lower bound and a heuristic to compute an upper bound. Non-optimal arcs and nodes were identified and eliminated based on the reduced costs. This method was tested on a set of randomly generated instances, and the results showed that they could solve instances with up to 104 nodes and 8 clusters. Fischetti *et al.* (1997) proposed an efficient branch-and-cut algorithm to solve the AGTSP. They developed exact and heuristic separation procedures for some classes of facet-defining inequalities. They also generated a library of GTSP instances called GTSP-LIB by taking TSP-LIB instances and performing a clustering procedure on the nodes. Their algorithm could solve instances with up to 89 clusters and 442 nodes.

A different approach to solve the GTSP is to develop heuristics. Gutin & Karapetyan (2010) proposed a memetic algorithm combining genetic and powerful local search algorithms. They reported excellent results on the GTSP-LIB instances, with computation times less than 60 seconds and most of the solutions within 0.2% of the best-known values. Helsgaun (2015) extended the Lin-Kernighan-Helsgaun TSP heuristic (Helsgaun, 2000, 2009) to the GTSP. The resulting algorithm improved the solution quality on GTSP-LIB instances compared with the memetic algorithm proposed in Gutin & Karapetyan (2010), at the expense of more computation time. Smith & Imeson (2017) developed an algorithm based on adaptive large neighborhood search. Their results showed that given the same amount of computation time, their algorithm was competitive on instances from the GTSP-LIB and other libraries.

The GTSPTW is also related to the TSPTW. When all the clusters of the GTSPTW are singletons, i.e., they contain only one vertex, the problem reduces to the TSPTW. Ascheuer *et al.* (2001) proposed several formulations for the asymmetric TSPTW and compared them within a branch-and-cut scheme. They incorporated techniques such as data pre-processing, primal heuristics, local search and variable fixing, in addition to separation algorithms. Dash *et al.* (2012) presented an extended formulation for the TSPTW based on the partitioning of the TW into sub-windows, which they called buckets. The LP relaxation of this formulation provided strong lower bounds. Strong valid inequalities (bucket sequential ordering polytope inequalities) were generated and incorporated in a branch-and-cut framework. Their results showed that the proposed formulation was effective and solved several previously unsolved benchmark instances. The state-of-the-art exact algorithm for the TSPTW is the dynamic programming algorithm proposed by Baldacci *et al.* (2012). Three different state space relaxations were used to obtain tight lower bounds and were combined with a dynamic programming algorithm. This method can solve all but one instance from the literature to optimality.

Although there is no article related to the GTSPTW, one can find related papers in the multi-vehicle case. Ghiani & Improta (2000) extended the GTSP to the generalized vehicle routing problem (GVRP) by introducing quantities to be delivered to customers and considering vehicles with limited capacity. Only a few works address the multi-vehicle case with TWs, but with some restrictions on the TWs. Moccia *et al.* (2012) proposed a tabu search method for what they called the Generalized-VRPTW. However, they define a TW for each cluster while a TW is associated with every vertex in the GTSPTW. They proved the effectiveness of the method by testing it on GVRP instances and multi-depot VRPTW instances. Recently, Reyes *et al.* (2017) examined the special case where TWs within the same cluster do not overlap. They were inspired by the trunk delivery system we mentioned in Section 1 and considered what they called the VRP with Roaming Delivery Locations (VRPRDL). The authors developed construction and improvement heuristics for the problem, and their results illustrated the advantage of applying the trunk delivery over the traditional home delivery. Following this work, Ozbaygin *et al.* (2017) formulated VRPRDL as a set-partitioning problem and proposed a branch-and-price algorithm. Moreover, they came up with a hybrid delivery strategy combining trunk delivery and home delivery, in which case the TWs

within a cluster are no longer non-overlapping. Their results revealed that employing this strategy led to an average savings of nearly 20% compared with the standard delivery system when only home delivery is used.

3.3 Problem definition and mathematical modeling

The GTSPTW can be formally defined as follows: given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, the set of vertices $\mathcal{V} = \{0, 1, ..., N\}$ is partitioned into $\mathcal{C}_0 = \{0\}, \mathcal{C}_1, ..., \mathcal{C}_K$ clusters. $\mathcal{K} = \{0, 1, ..., K\}$ denotes the cluster index set. Cluster $\mathcal{C}_0$ contains only the starting and ending vertex, i.e. the depot. A TW $[E_i, L_i]$, is associated with each vertex $i \in \{0, 1, ..., N\}$ with $[E_0, L_0] = [0, T]$ representing the optimization time horizon. A visit can only be made to a vertex during its TW, and an early arrival leads to waiting time while a late arrival causes infeasibility. There is no assumption on the TWs for a given cluster, i.e., TWs can overlap or be disjointed. Arcs are only defined between vertices belonging to different clusters, that is, $\mathcal{A} \subseteq \{(i, j) | i \in \mathcal{C}_k, j \in \mathcal{C}_l, k \neq l, k, l \in \mathcal{K}\}$. A traveling cost C_{ij} and a traveling time T_{ij} are associated with each arc $(i, j) \in \mathcal{A}$. We call an arc (i, j) *feasible* if $E_i + T_{ij} \leq L_j$, which means that vertex j can be reached from vertex i through arc (i, j). $\mathcal{A}$ is defined as the set of *feasible* arcs, which is a subset of $\{(i, j) | i \in \mathcal{C}_k, j \in \mathcal{C}_l, k \neq l, k, l \in \mathcal{K}\}$.

The objective of the GTSPTW is to find a minimum cost tour starting and ending at the depot such that each cluster is visited exactly once and the TW constraints are respected, i.e., one vertex of each cluster is visited during its time window.

Let us introduce the following notation. For any set $\mathcal{S} \subset \mathcal{V}$, $\delta^+(\mathcal{S}) = \{(i, j) \in \mathcal{A} | i \in \mathcal{S}, j \notin \mathcal{S}\}$, $\delta^-(\mathcal{S}) = \{(i, j) \in \mathcal{A} | i \notin \mathcal{S}, j \in \mathcal{S}\}$. For simplicity, when $\mathcal{S} = \{i\}$, we use $\delta^+(i)$ and $\delta^-(i)$ as opposed to $\delta^+(\{i\})$ and $\delta^-(\{i\})$.

To model the GTSPTW, we define three set of variables. For all $(i, j) \in \mathcal{A}$, let x_{ij} be a binary variable equal to one if and only if arc $(i, j) \in \mathcal{A}$ belongs to the tour. For all $i \in \mathcal{V}$, let y_i be a binary variable equal to one if $i \in \mathcal{V}$ belongs to the tour, and t_i be the service time at vertex $i \in \mathcal{V}$. For the depot, the service time actually corresponds to the departure time. A first compact mathematical programming formulation $\mathcal{F}1$ is as follows:

$$(\mathcal{F}1) \quad \text{minimize} \quad \sum_{(i,j) \in \mathcal{A}} C_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{C}_k} y_i = 1 \qquad \forall k \in \mathcal{K}, \tag{3.2}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V}, \qquad (3.3)$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i \qquad \forall i \in \mathcal{V}, \qquad (3.4)$$

$$E_i y_i \leq t_i \leq L_i y_i \qquad \forall i \in \mathcal{V}, \qquad (3.5)$$

$$t_i - t_j + T_{ij} x_{ij} \leq L_i y_i - E_j y_j - (L_i - E_j) x_{ij} \qquad \forall (i,j) \in \mathcal{A}, j \neq 0, \qquad (3.6)$$

$$t_i + T_{i0} x_{i0} \leq L_0 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \qquad (3.7)$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{V}, \qquad (3.8)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, \qquad (3.9)$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{V}. \qquad (3.10)$$

The objective function (3.1) minimizes the total traveling cost. Constraints (3.2) ensure that exactly one vertex from each cluster is visited. Constraints (3.3) and (3.4) are flow conservation constraints. Constraints (3.5) ensure that each vertex is visited during its TW. Constraints (3.6) ensure that the service times are consistent. If vertex j is visited just after vertex i, then constraint (3.6) will ensure $t_j \geq t_i + T_{ij}$. In addition, constraints (3.6) eliminate subtours since they generalize the subtour elimination constraints of Miller-Tucker-Zemlin for the traveling salesman problem (Miller *et al.*, 1960). Constraints (3.7) ensure that the tour ends at the depot before its TW closes. Constraints (3.8)∼(3.10) are related to variable definitions.

The second formulation is based on the following observation. Since only one vertex is selected in each cluster, we can define one time variable per cluster instead of defining a time variable for every vertex as above. Let $\tau_k \geq 0$, $k \in \mathcal{K}$ be the service time at cluster k.

In the second formulation $\mathcal{F}2$, the objective function and constraints (3.2)∼(3.4), (3.8)∼(3.9) are as in $\mathcal{F}1$. Constraints (3.5)∼(3.7) and (3.10) are replaced by (3.11)∼(3.13) and (3.14) respectively. We obtain the following compact model:

($\mathcal{F}2$) minimize (3.1)

s.t. (3.2) ∼ (3.4)

$$\sum_{i\in\mathcal{C}_k} E_i y_i \leq \tau_k \leq \sum_{i\in\mathcal{C}_k} L_i y_i \qquad \forall k \in \mathcal{K}, \qquad (3.11)$$

$$\tau_h - \tau_k + \sum_{\substack{i\in\mathcal{C}_h \\ j\in\mathcal{C}_k}} T_{ij} x_{ij} \leq \sum_{i\in\mathcal{C}_h} L_i y_i - \sum_{j\in\mathcal{C}_k} E_j y_j$$

$$- \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} (L_i - E_j) x_{ij} \quad \forall h \in \mathcal{K},\ k \in \mathcal{K} \setminus \{0\}, h \neq k, \quad (3.12)$$

$$\tau_k + \sum_{i \in \mathcal{C}_k} T_{i0} x_{i0} \leq L_0 \quad \forall k \in \mathcal{K} \setminus \{0\}, \quad (3.13)$$

(3.8) $\sim$ (3.9)

$$\tau_k \geq 0 \quad \forall k \in \mathcal{K}. \quad (3.14)$$

Note that using constraints (3.3) or constraints (3.4) the y variables can be substituted by $\sum_{(i,j)\in\delta^+(i)} x_{ij}$ or by $\sum_{(j,i)\in\delta^-(i)} x_{ji}$. Thus, they may be omitted in both formulations $\mathcal{F}1$ and $\mathcal{F}2$.

3.4 Valid inequalities for the GTSPTW

In this section, we present some inequalities we developed for GTSPTW, mainly adapted from valid inequalities for the Steiner tree problem, the GTSP or the TSPTW. Some of these inequalities are defined on x variables only, other inequalities also involve y variables. We also present lifted versions of some valid inequalities. Section 3.4.1 proposes a lifted version of the Miller-Tucker-Zemlin (MTZ) inequalities provided in formulations $\mathcal{F}1$ and $\mathcal{F}2$. Sections 3.4.2 and 3.4.3 propose polynomial-size families of valid inequalities while sections 3.4.4 to 3.4.7 provide exponential-size families of valid inequalities.

3.4.1 Supervalid MTZ inequalities

Desrochers & Laporte (1991) observed that the subtour elimination constraints presented in the MTZ version as in (3.6) can be lifted by taking the reverse arcs $(j,i) \in \mathcal{A}$ into account. Moreover, it can be noticed that since waiting times are allowed, formulation $\mathcal{F}1$ may have multiple optimal solutions, i.e., given optimal values for the x_{ij} variables, there may be several values for the t_i variables that satisfy constraints (3.5)$\sim$(3.7). In addition, given optimal values for x_{ij} variables, there are always feasible values for the t_i variables such that each vertex is visited as early as possible, namely minimizing waiting times. For routing problems with time windows, Yuan *et al.* (2019b) proposed the so-called supervalid MTZ inequalities, taking the reverse arcs into consideration and ensuring vertices are visited as soon as possible. An inequality is *supervalid* if it does not cut off all optimal solutions. This concept is a

generalization of the concept of valid inequalities and has been introduced by Israeli & Wood (2002). We adapt the supervalid MTZ inequalities proposed by Yuan *et al.* (2019b) for the GTSPTW as follows.

Proposition 3.1. *For $i, j \in \mathcal{V} \setminus \{0\}$ suppose that arcs (i, j) and $(j, i) \in \mathcal{A}$ are feasible, then*

$$t_i - t_j + T_{ij}x_{ij} + min\{-T_{ji}, E_j - E_i\}x_{ji} \leq L_i y_i - E_j y_j - (L_i - E_j)(x_{ij} + x_{ji}) \quad (3.15)$$

are supervalid inequalities for formulation $\mathcal{F}1$.

Proof. See Appendix A. □

Similarly, for formulation $\mathcal{F}2$ in which the time variables are defined for clusters, we can adapt the supervalid MTZ constraints as follows.

Proposition 3.2. *For $h, k \in \mathcal{K} \setminus \{0\}, h \neq k$ suppose that there exists $i \in \mathcal{C}_h$ and $j \in \mathcal{C}_k$ such that the arcs (i, j) and (j, i) are both feasible. Then*

$$\begin{aligned} &\tau_h - \tau_k + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} T_{ij}x_{ij} + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} min\{-T_{ji}, E_j - E_i\}x_{ji} \\ &\leq \sum_{i \in \mathcal{C}_h} L_i y_i - \sum_{j \in \mathcal{C}_k} E_j y_j - \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} (L_i - E_j)(x_{ij} + x_{ji}) \end{aligned} \quad (3.16)$$

are supervalid inequalities for formulation $\mathcal{F}2$.

Proof. See Appendix A. □

3.4.2 Arc selection inequalities

It is obvious that in any feasible solution at most one arc (i, j) or (j, i) is selected. Therefore, we have:

$$x_{ij} + x_{ji} \leq y_i \qquad \forall i, j \in \mathcal{V} \text{ such that } (i, j), (j, i) \in \mathcal{A}. \quad (3.17)$$

Since only one vertex is selected from each cluster, we can lift inequalities (3.17) and obtain:

Proposition 3.3. *The following constraints are valid inequalities for the GTSPTW:*

$$\sum_{j \in \mathcal{C}_k} x_{ij} + \sum_{j \in \mathcal{C}_k} x_{ji} \leq y_i \qquad \forall i \in \mathcal{V}, k \in \mathcal{K}, i \notin \mathcal{C}_k \text{ and } \exists j \in \mathcal{C}_k \text{ such that } (i, j), (j, i) \in \mathcal{A}. \quad (3.18)$$

3.4.3 Arc-or-vertex inequalities

In a feasible solution, a vertex and an arc may not be simultaneously present due to TW constraints. We introduce *arc-or-vertex inequalities* to exploit this property and these inequalities impose that at most one of the vertex or the arc is chosen. Let $\mathcal{C}_k^{ij}$ be the subset of $\mathcal{C}_k$ containing vertices that cannot be visited before or after arc (i, j). A vertex $h \in \mathcal{C}_k$ belongs to $\mathcal{C}_k^{ij}$ if there is no feasible path going from h to j including (i, j) and there is no feasible path going from i to h traversing (i, j). Figure 3.2 depicts this situation. Formally, a vertex h belongs to $\mathcal{C}_k^{ij}$ if and only if:

1. $h \in \mathcal{C}_k$,
2. $E_h + SP_{hi} > L_i$, or $E_h + SP_{hi} + T_{ij} > L_j$,
3. $E_j + SP_{jh} > L_h$, or $E_i + T_{ij} + SP_{jh} > L_h$.

where SP_{ij} is the shortest traveling time between vertices i and j. When the triangle inequality is not satisfied for the traveling time matrix (i.e., $T_{il} + T_{lj} < T_{ij}$ for at least a triplet $i, j, l \in \mathcal{V}$), the shortest traveling time SP_{ij} going from vertex i to vertex j can include the visit of other vertices and can be lower than T_{ij}.

Proposition 3.4. *The following constraints are valid inequalities for the GTSPTW:*

$$x_{ij} + \sum_{h \in \mathcal{C}_k^{ij}} y_h \leq 1 \qquad \forall (i, j) \in \mathcal{A}, i, j \neq 0, \forall k \in \mathcal{K}, i, j \notin \mathcal{C}_k. \tag{3.19}$$

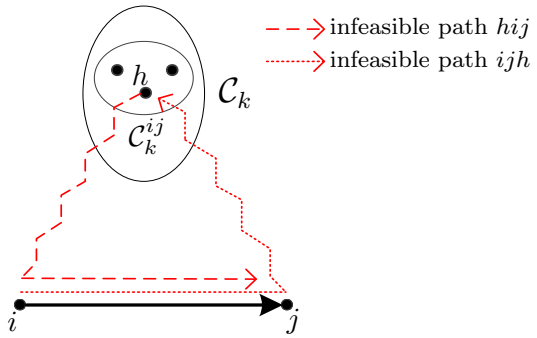


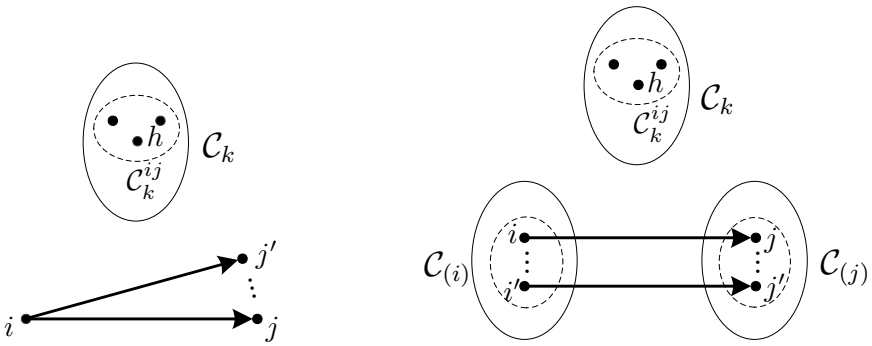
Figure 3.2: Arc-or-vertex inequalities example.

We can lift the arc-or-vertex inequalities in two different ways as depicted in Figure 3.3. First, given an arc $(i, j) \in \mathcal{A}$ and a cluster $k \in \mathcal{K}$ with a set of incompatible vertices $\mathcal{C}_k^{ij}$, other arcs (i, j') may lead to the same set of incompatible vertices:

$\mathcal{C}_k^{ij'} = \mathcal{C}_k^{ij}$. In this case, constraints (3.19) can be lifted by summing up over such variables $x_{i,j'}$ since at most one of these arcs can be present in a feasible solution. Thus, inequalities (3.19) can be strengthened as follows:

Proposition 3.5. *For all $(i,j) \in \mathcal{A}, i,j \neq 0$ and for all $k \in \mathcal{K}$ such that $i,j \notin \mathcal{C}_k$, consider $\mathcal{V}_k^{ij} = \{j' \in \mathcal{V} \setminus \mathcal{C}_k \mid \mathcal{C}_k^{ij'} = \mathcal{C}_k^{ij}\}$, then constraints (3.19) can be lifted as:*

$$\sum_{j' \in \mathcal{V}_k^{ij}} x_{ij'} + \sum_{h \in \mathcal{C}_k^{ij}} y_h \leq 1 \tag{3.20}$$



(a) Lift 1 (b) Lift 2

Figure 3.3: Lifting of arc-or-vertex inequalities.

Second, given an arc $(i,j) \in \mathcal{A}$ and a cluster $k \in \mathcal{K}$ with a set of incompatible vertices $\mathcal{C}_k^{ij}$, other arcs (i',j') may lead to the exactly same set of incompatible vertices: $\mathcal{C}_k^{i'j'} = \mathcal{C}_k^{ij}$. When i and i' belong to the same cluster, respectively j and j' belong to the same cluster, then constraints (3.19) can be lifted by summing up over such variables $x_{i',j'}$ since such arcs are defined between the same pair of clusters and at most one appears in any feasible solution.

Proposition 3.6. *Let $\mathcal{C}_{(i)}$ be the cluster containing vertex i. For all $(i,j) \in \mathcal{A}, i,j \neq 0$ and for all $k \in \mathcal{K}$ such that $i,j \notin \mathcal{C}_k$, consider $\mathcal{A}_k^{ij} = \{(i',j') \in \mathcal{A} \mid i' \in \mathcal{C}_{(i)}, j' \in \mathcal{C}_{(j)}, \mathcal{C}_k^{i'j'} = \mathcal{C}_k^{ij}\}$, then inequalities (3.19) can be strengthened as:*

$$\sum_{(i',j') \in \mathcal{A}_k^{ij}} x_{i'j'} + \sum_{h \in \mathcal{C}_k^{ij}} y_h \leq 1 \tag{3.21}$$

3.4.4 Generalized subtour elimination constraints

Although constraints (3.6) eliminate tours not including the depot in any feasible integer solution, subtours may be present when the integer requirement on variables x

and y is relaxed. Thus, the subtour elimination constraints (SEC) defined by Dantzig *et al.* (1954) for the TSP can increase the linear relaxation value. These constraints can be generalized to take into account the presence of clusters (Fischetti *et al.*, 1997). For subsets of clusters, they can be expressed as follows:

Proposition 3.7. *The constraints*

$$\sum_{(i,j)\in\delta^+(\mathcal{S})} x_{ij} \geq 1 \qquad \forall \mathcal{S} = \cup_{h\in\mathcal{H}}\mathcal{C}_h,\ \mathcal{H}\subset\mathcal{K},\ 2\leq|\mathcal{H}|\leq K-1. \tag{3.22}$$

are valid for the GTSPTW.

Note that if we consider constraints (3.18) defined for $i \in \mathcal{C}_l, k \in \mathcal{K}$ and we sum these constraints over all $i \in \mathcal{C}_l$, we obtain:

$$\sum_{i\in\mathcal{C}_l}(\sum_{j\in\mathcal{C}_k} x_{ij} + \sum_{j\in\mathcal{C}_k} x_{ji}) \leq 1 \tag{3.23}$$

which prevents subtours of length two between vertices of different clusters.

3.4.5 SOP inequalities

SOP inequalities are based on the notion of precedence between pairs of vertices and were introduced by Balas *et al.* (1995) in the context of the precedence-constrained Asymmetric Traveling Salesman Problem (ATSP), also known as the Sequential Ordering Problem (SOP). These inequalities are also effective for TSPTW where precedences between nodes are inferred based on the TW restrictions (Ascheuer *et al.*, 2001; Dash *et al.*, 2012). Here we extend the SOP inequalities to the GTSPTW.

Recall that $\mathcal{C}_{(i)}$ denotes the cluster containing vertex i. We say that a vertex $i \in \mathcal{V}$ *precedes* vertex $j \in \mathcal{V}\setminus\mathcal{C}_{(i)}$ if i has to be visited before j in any feasible solution. We denote this relation as $i \prec j$. When the triangle inequality for the traveling time matrix is satisfied, the precedence between two vertices i and j is defined as:

$$i \prec j \text{ if } E_i + T_{ij} \leq L_j \text{ and } E_j + T_{ji} > L_i. \tag{3.24}$$

The relation (3.24) can be extended to the case where the triangle inequality is not satisfied. Then, we consider SP_{ij} the shortest traveling time from vertex i to vertex j. The path with the shortest traveling time may include other vertices since the triangle inequality is not satisfied. Thus, the precedence relation becomes:

$$i \prec j \text{ if } E_i + SP_{ij} \leq L_j \text{ and } E_j + SP_{ji} > L_i. \tag{3.25}$$

If the triangle inequality is satisfied, both relations (3.25) and (3.24) are equivalent since $SP_{ij} = T_{ij}$.

For any given $i \in \mathcal{V} \setminus \{0\}$, we define $\pi(i) = \{j \in \mathcal{V} \setminus \mathcal{C}_{(i)} : j \prec i\}$ the set of vertices that precede vertex i, and $\sigma(i) = \{j \in \mathcal{V} \setminus \mathcal{C}_{(i)} : i \prec j\}$ the set of vertices that succeed vertex i. In the following, we summarize the classes of SOP inequalities we used in our implementation. For the ease of explanation, we introduce the following notations. For any two vertex sets $\mathcal{U}, \mathcal{W} \subseteq \mathcal{V}$, let $(\mathcal{U} : \mathcal{W}) = \{(i,j) \in \mathcal{A} | i \in \mathcal{U}, j \in \mathcal{W}\}$. This set correspond to the cut between the two vertex sets $\mathcal{U}$ and $\mathcal{W}$. Let us use $x(\mathcal{U} : \mathcal{W})$ to indicate $\sum_{i \in \mathcal{U}, j \in \mathcal{W}} x_{ij}$. For any set of vertices $\mathcal{S} \subseteq \mathcal{V}$, we note $\bar{\mathcal{S}} = \mathcal{V} \setminus \mathcal{S}$ its complement.

Proposition 3.8. *For $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $i \in \mathcal{S}$ such that $\pi(i) \neq \emptyset$, the predecessor inequalities (π-inequalities):*

$$x((\mathcal{S} \setminus \pi(i)) : (\bar{\mathcal{S}} \setminus \pi(i))) \geq y_i \tag{3.26}$$

are valid for GTSPTW. The π-inequalities (3.26) can be strengthened as:

$$x((\mathcal{S} \setminus \mathcal{P}_i) : (\bar{\mathcal{S}} \setminus \mathcal{P}_i)) \geq y_i \tag{3.27}$$

where $\mathcal{P}_i = \pi(i) \cup \mathcal{C}_{(i)} \setminus \{i\}$.

Proof. If $y_i = 0$, then the inequality is obviously verified. When $y_i = 1$, vertex i is visited in tour $\mathcal{T}$ representing a feasible solution. Let $\hat{s}$ be the last vertex of $\mathcal{S}$ visited by $\mathcal{T}$. Since $i \in \mathcal{S}$, $\hat{s} \notin \pi(i)$, so $\hat{s} \in \mathcal{S} \setminus \pi(i)$. Moreover, the successor $\hat{t}$ of $\hat{s}$ in tour $\mathcal{T}$ cannot be in $\pi(i)$, so $\hat{t} \in \bar{\mathcal{S}} \setminus \pi(i)$. Clearly, any feasible tour $\mathcal{T}$ contains at least one arc going from $\mathcal{S} \setminus \pi(i)$ to $\bar{\mathcal{S}} \setminus \pi(i)$. The strengthening can be deduced from the observation that only one vertex per cluster is visited in any feasible solution of the GTSPTW. □

Similarly, we can straightforwardly establish the successor inequalities (*σ-inequalities*) for the GTSPTW and strengthen them as follows.

Proposition 3.9. *For $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $j \in \mathcal{S}$ such that $\sigma(j) \neq \emptyset$, the σ-inequalities:*

$$x((\bar{\mathcal{S}} \setminus \sigma(j)) : (\mathcal{S} \setminus \sigma(j))) \geq y_j. \tag{3.28}$$

are valid for GTSPTW. The σ-inequalities (3.28) can be strengthened as:

$$x((\bar{\mathcal{S}} \setminus \mathcal{Q}_j) : (\mathcal{S} \setminus \mathcal{Q}_j)) \geq y_j \tag{3.29}$$

where $\mathcal{Q}_j = \sigma(j) \cup \mathcal{C}_{(j)} \setminus \{j\}$.

For vertices $i \in \mathcal{V} \setminus \{0\}, j \in \mathcal{V}$ such that $i \prec j$, we can establish the predecessor-successor inequalities ((π,σ)-inequalities) and strengthen them as follows.

Proposition 3.10. *For $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $i \in \mathcal{S}$, $j \in \bar{\mathcal{S}}$ such that $i \prec j$, $\pi(i) \neq \emptyset$ and $\sigma(j) \neq \emptyset$, the (π,σ)-inequalities:*

$$x((\mathcal{S} \setminus (\pi(i) \cup \sigma(j))) : (\bar{\mathcal{S}} \setminus (\pi(i) \cup \sigma(j)))) \geq y_i + y_j - 1. \tag{3.30}$$

are valid for GTSPTW. The (π, σ)-inequalities (3.30) can be strengthened as:

$$x((\mathcal{S} \setminus \mathcal{W}_{ij}) : (\bar{\mathcal{S}} \setminus \mathcal{W}_{ij})) \geq y_i + y_j - 1 \tag{3.31}$$

where $\mathcal{W}_{ij} = \pi(i) \cup \sigma(j) \cup \mathcal{C}_{(i)} \cup \mathcal{C}_{(j)} \setminus \{i, j\}$.

Proof. If $y_i + y_j \leq 1$, then the inequality is obviously verified. When $y_i = y_j = 1$, vertex i and j are present in tour $\mathcal{T}$ representing a feasible solution. Since $i \prec j, i \in \mathcal{S}, j \in \bar{\mathcal{S}}$, then $i \in \mathcal{S} \setminus (\pi(i) \cup \sigma(j))$ and $j \in \bar{\mathcal{S}} \setminus (\pi(i) \cup \sigma(j))$. Since $i \prec j$, it is obvious that any feasible tour $\mathcal{T}$ contains at least one arc going from $\mathcal{S} \setminus (\pi(i) \cup \sigma(j))$ to $\bar{\mathcal{S}} \setminus (\pi(i) \cup \sigma(j))$. □

3.4.6 SOP inequalities defined on clusters

In Section 3.4.5 we presented the SOP inequalities based on the precedence relationship between vertices. We can also define the precedence between a vertex and a cluster.

Let us denote by $i \prec \mathcal{C}_k$ (resp. $\mathcal{C}_k \prec i$) the precedence relation between a vertex and a cluster, i.e., $i \prec \mathcal{C}_k$ if and only if $i \prec j, \forall j \in \mathcal{C}_k$ (resp. $\mathcal{C}_k \prec i$ if and only if $j \prec i, \forall j \in \mathcal{C}_k$). It follows that, if vertex i belongs to a solution, then it has to be visited before (resp. after) any vertex of cluster $\mathcal{C}_k$. Let us indicate by $\mathcal{C}_h \prec \mathcal{C}_k$ the precedence relation between two clusters, i.e., $\mathcal{C}_h \prec \mathcal{C}_k$ if and only i $i \prec j, \forall i \in \mathcal{C}_h, j \in \mathcal{C}_k$. We define $\pi(\mathcal{C}_k) = \{i \in \mathcal{V} \setminus \mathcal{C}_k : i \prec \mathcal{C}_k\}$, $\sigma(\mathcal{C}_k) = \{i \in \mathcal{V} \setminus \mathcal{C}_k : \mathcal{C}_k \prec i\}$. Then we can extend the SOP inequalities as follows.

Proposition 3.11. *Let $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $\mathcal{C}_k \subseteq \mathcal{S}$, $\pi(\mathcal{C}_k) \neq \emptyset$, the $\pi_{\mathcal{C}_k}$-inequalities:*

$$x((\mathcal{S} \setminus \pi(\mathcal{C}_k)) : (\bar{\mathcal{S}} \setminus \pi(\mathcal{C}_k))) \geq 1. \tag{3.32}$$

are valid for the GTSPTW.
Let $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $\mathcal{C}_k \subseteq \mathcal{S}$, $\sigma(\mathcal{C}_k) \neq \emptyset$, the $\sigma_{\mathcal{C}_k}$-inequalities:

$$x((\bar{\mathcal{S}} \setminus \sigma(\mathcal{C}_k)) : (\mathcal{S} \setminus \sigma(\mathcal{C}_k))) \geq 1. \tag{3.33}$$

are valid for the GTSPTW.
Let $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $\mathcal{C}_h \prec \mathcal{C}_k$, $\mathcal{C}_h \subseteq \mathcal{S}, \mathcal{C}_k \subseteq \bar{\mathcal{S}}$, $\pi(\mathcal{C}_h) \neq \emptyset, \sigma(\mathcal{C}_k) \neq \emptyset$, the $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequalities:

$$x((\mathcal{S} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k))) : (\bar{\mathcal{S}} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k)))) \geq 1. \tag{3.34}$$

are valid for the GTSPTW.

Proof. See Appendix A. □

3.4.7 Clique inequalities

Let $\mathcal{S} \subset \mathcal{V}$ be a subset of vertices. Due to the presence of TWs, there may not exist a feasible path visiting all vertices of $\mathcal{S}$. In this case we say that $\mathcal{S}$ is *infeasible*. Then the number of visited vertices in such $\mathcal{S}$ in any feasible solution must be strictly less than $|\mathcal{S}|$ (Padberg, 1973). The clique inequalities can be expressed as follows:

$$\sum_{i \in \mathcal{S}} y_i \leq |\mathcal{S}| - 1. \tag{3.35}$$

These inequalities can also be lifted.

Proposition 3.12. *Let us consider $\mathcal{S} \subset \mathcal{V}$ such that $\mathcal{S}$ is infeasible and $|\mathcal{S} \cap \mathcal{C}_k| \leq 1, \forall k \in \mathcal{K}$. For $j \in \mathcal{S}$, let us define $\mathcal{S}(j) = \{i \in \mathcal{C}_{(j)} \setminus \{j\} | \mathcal{S}' = (\mathcal{S} \setminus \{j\}) \cup \{i\}$ is infeasible$\}$. If $|\mathcal{S}(j)| \neq 0$, the clique inequalities (3.35) can be strengthened as:*

$$\sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{S}(j)} y_i \leq |\mathcal{S}| - 1 \qquad \forall j \in \mathcal{S}. \tag{3.36}$$

For $h \in \mathcal{S} \setminus \{j\}$, let us define $\mathcal{S}_j(h) = \{i \in \mathcal{C}_{(h)} \setminus \{h\} | (\mathcal{S} \cup \{j^*, i\}) \setminus \{j, h\}$ is infeasible, $\forall j^* \in \mathcal{S}(j) \cup \{j\}\}$. If $|\mathcal{S}_j(h)| \neq 0$, inequalities (3.36) can be lifted as:

$$\sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{S}(j)} y_i + \sum_{i \in \mathcal{S}_j(h)} y_i \leq |\mathcal{S}| - 1 \qquad \forall j, h \in \mathcal{S}. \tag{3.37}$$

Proof. See Appendix A. □

3.5 The branch-and-cut algorithm

In this section, we describe the branch-and-cut algorithm we propose to solve the GTSPTW. The algorithm consists of three main phases. The first phase is the preprocessing step that is invoked before starting the optimization procedure. It is presented in Section 3.5.1. Then we apply a quick heuristic to obtain a feasible solution and to provide an upper bound of the optimal value. Details are given in Section 3.5.2. Finally, the main phase consists in solving the problem by using a branch-and-cut algorithm, based on the standard branch-and-cut scheme provided by the commercial solver CPLEX 12.6.3. The initial model is built based on the mixed integer linear programming formulations $\mathcal{F}1$ or $\mathcal{F}2$ with the strengthened MTZ-inequalities proposed in Section 3.4.1. The initial solution obtained by the heuristic is used as a warm start for the branch-and-cut procedure. Inside the branch-and-bound tree, each time a fractional solution is obtained, valid inequalities proposed in Section 3.4 are checked, and

the ones violated by the current solution are added to the model. Details are given in Section 3.5.3. For valid inequalities with a polynomial number of constraints, we memorize all of them and scan the entire set to seek for those that are violated (see Section 3.5.3.3). For exponential-size families of constraints (GSEC and SOP inequalities), separation algorithms are applied to efficiently detect the violated inequalities (see Sections 3.5.3.1 and 3.5.3.2), and the number of the inequalities we choose to separate is limited (see Section 3.5.3.4).

3.5.1 Data preprocessing

As with many other combinatorial optimization problems, preprocessing is an important feature to enhance the resolution of the problem. In our case, preprocessing step consists in tightening the TWs and eliminating arcs that cannot be part of a feasible solution.

The TW width can be reduced by taking into account the earliest and the latest arrival and departure times at each vertex of the graph from or to another vertex. In particular, we consider the following conditions proposed by Desrochers *et al.* (1992):

- earliest arrival time from predecessors: $E_i = \max\{E_i, \min\{L_i, \min_{(j,i)\in\mathcal{A}}(E_j + T_{ji})\}\}$;
- earliest departure time to successors: $E_i = \max\{E_i, \min\{L_i, \min_{(i,j)\in\mathcal{A}}(E_j - T_{ij})\}\}$;
- latest arrival time from predecessors: $L_i = \min\{L_i, \max\{E_i, \max_{(j,i)\in\mathcal{A}}(L_i + T_{ji})\}\}$;
- latest departure time to successors: $L_i = \min\{L_i, \max\{E_i, \max_{(i,j)\in\mathcal{A}}(L_j - T_{ij})\}\}$.

These conditions are applied iteratively to all vertices until no TW can be reduced. After applying the TW reduction, we sparsify the graph by eliminating the arcs that cannot be part of a feasible solution. In particular, we remove:

- arcs $(i,j) \in \mathcal{A}$ such that $E_i + T_{ij} > L_j$;
- arcs $(i,j) \in \mathcal{A}, i,j \neq 0$ such that $\exists\, k \in \mathcal{K}$, $\forall h \in \mathcal{C}_k$, both $E_h + SP_{hi} + T_{ij} > L_j$ and $E_i + T_{ij} + SP_{jh} > L_h$ hold, i.e., arc (i,j) can be traversed neither before nor after visiting one cluster $\mathcal{C}_k$.

This results in the elimination of the corresponding x variables in the formulation.

3.5.2 Initial heuristic

We develop a simple and fast heuristic to identify a feasible solution for the GTSPTW. The solution obtained is used as a warm start in the branch-and-cut procedure.

First, we extend the refinement procedure for GTSP proposed by Fischetti *et al.* (1997) to GTSPTW case. Suppose we have a visiting sequence $(h_1, \ldots, h_p)$ of p different clusters in $\mathcal{K} \setminus \{0\}$. Based on this sequence we construct a layered network (LN) as depicted in Figure 4.2. This network has $p+2$ layers corresponding to clusters $C_{h_0} = \mathcal{C}_0, \mathcal{C}_{h_1}, \ldots, \mathcal{C}_{h_p}, \mathcal{C}_{h_{p+1}} = \mathcal{C}_0$, with their respective vertices. Clusters $\mathcal{C}_{h_0}$ and $\mathcal{C}_{h_{p+1}}$ both represent the depot. The LN contains arcs $(i,j) \in \mathcal{A}$ such that $i \in \mathcal{C}_{h_f}, j \in \mathcal{C}_{h_{f+1}}, f = 0, \ldots, p$. The objective is to find a path in the LN that starts at $\mathcal{C}_{h_0}$ and ends at $\mathcal{C}_{h_{p+1}}$ visiting exactly one vertex of each layer, that is, one vertex from each cluster. The solution can be found by determining the shortest path with TWs from $\mathcal{C}_{h_0}$ to $\mathcal{C}_{h_{p+1}}$. If $p = K$, the resulting path (if it exists) provides a feasible GTSPTW solution.

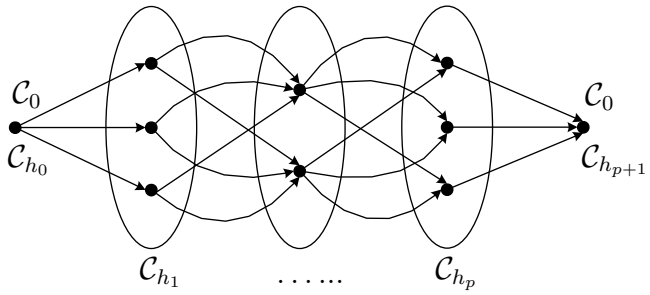


Figure 3.4: The layered network.

A labeling algorithm is applied to determine the shortest path with TWs on the LN. A label L_i associated with a vertex i consists of a pair (C_i, T_i) representing respectively the cost and service time of a feasible partial path that starts at $\mathcal{C}_{h_0}$ and arrives at vertex i. Let $\mathcal{L}(i)$ be the set containing all the labels associated with vertex i. Suppose that $\mathcal{C}_{cur}$ is the current cluster and $\mathcal{C}_{pre}$ is the previous one. First we compute the label set $\mathcal{L}(i)$, for all $i \in \mathcal{C}_{cur}$ by extending labels in $\mathcal{L}(j)$, for all $j \in \mathcal{C}_{pre}$. Extending a label $L_j \in \mathcal{L}(j)$ towards a vertex $i \in \mathcal{C}_{cur}$ consists in creating another label $L_i \in \mathcal{L}(i)$ such that:

$$C_i = C_j + C_{ij}; \tag{3.38}$$

$$T_i = \max\{E_i, T_j + T_{ji}\}. \tag{3.39}$$

If $T_i > L_i$, the partial path associated with the label is infeasible and this label is disregarded.

To make the algorithm efficient, we only keep non-dominated labels. A label L_i^1 dominates a label L_i^2 if and only if $C_i^1 \leq C_i^2$ and $T_i^1 \leq T_i^2$. It is easy to see that extending L_i^1 on the same arcs to the last vertex of the LN would always produce a better solution than extending L_i^2 in the same way.

To generate good sequences of clusters, we develop the following constructive procedure. The initial sequence of clusters is empty, hence the corresponding LN contains two layers: $\mathcal{C}_{h_0} = \mathcal{C}_{h_1} = \mathcal{C}_0$. Then, at each step, we randomly select a cluster from those that are not yet inserted into the sequence. Suppose that the current sequence is $(h_1, \ldots, h_p)$, and cluster $\mathcal{C}_{h_l}$ is chosen to be inserted next. It is obvious that there are $p+1$ possible insertion positions for index h_l into the sequence. The labeling algorithm described above is invoked $p+1$ times, one for each candidate insertion, to determine the best insertion position. We record the sequence that provides the shortest path with TWs if such a sequence exists. If this is not the case, the sequence construction procedure is stopped. The cluster insertion procedure is repeated until $p = K$ to obtain a feasible solution for GTSPTW.

Algorithm 1 Heuristic to provide an initial solution.

```
R ← 2500
bestSol: best solution found
for h = 1 to R do
    H ← K \ {0} (the set of cluster index)
    feas ← true
    S ← ∅ (the sequence of clusters)
    while feas = true and H is not empty do
        l ← an index randomly chosen in set H
        Remove l from H
        Try to insert l at its best position in sequence S
        if the insertion of l is not feasible then
            feas ← false
    if feas = true then
        Update bestSol if S provides a better solution than the current bestSol
return bestSol
```

The sequence construction procedure is repeated R times, and the best solution is recorded. After preliminary experiments, we set $R = 2500$. A general description of the initial heuristic is provided in Algorithm 1. Note that when $K < 7$, there are only

720 possible cluster sequences. It is more efficient to enumerate all the sequences and to compute the shortest path with TWs for all of them. Since the labeling procedure is an exact procedure (given a sequence it finds the optimal path for that sequence), the optimal solution for the instance is found without calling the Branch-and-cut procedure. Therefore, the branch-and-cut algorithm is only applied for instances with $K \geq 7$ when applying the labeling procedure on all the sequences of size K becomes ineffective.

3.5.3 Separation techniques

At each node of the branch-and-cut tree, a relaxation of the model $\mathcal{F}1$ or $\mathcal{F}2$ is solved. Let us denote by $R\mathcal{F}1$ and $R\mathcal{F}2$ these relaxations. When solving a relaxed model, the binary requirement on the variables x and y is relaxed. Therefore, a fractional solution can be obtained where some values of the x or y variables are in the interval $]0, 1[$. Then, a separation algorithm is used to detect violated inequalities Since $\mathcal{F}1$ and $\mathcal{F}2$ are compact formulations of the GTSPTW, the valid inequalities proposed in Section 3.4 are not needed to define the problem but can help to strengthen the relaxed model. In the subsequent sections, we describe the separation algorithms that we implemented for the different families of valid inequalities. We indicate by (x^*, y^*) the current fractional solution.

3.5.3.1 Separation of the GSEC inequalities

It is well known that the separation of the SEC for the ATSP can be done by computing the maximum flow between the depot and each node j in the support graph $\mathcal{G}^*$ which corresponds to the undirected version of the original graph $\mathcal{G}$ where the capacity c_e of edge $e = \{i, j\}$ is equal to $x^*_{ij} + x^*_{ji}$. If each maximum flow is greater or equal than 2, the associated SEC is not violated by x^*. Otherwise, the minimum $(0 - j)$ cut induces a violated SEC inequality (Nemhauser & Wolsey, 1999).

To separate the GSEC (3.22), we consider a capacitated graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ where $\mathcal{V}^* = \mathcal{K}^* = \{0, 1, \ldots, K\}$, and $\mathcal{E}^* = \{\{k, l\} | k, l \in \mathcal{V}^*, k \neq l\}$. A capacity c_e is associated with each edge $e = \{k, l\} \in \mathcal{E}^*$, and defined as $c_e = \sum_{i \in \mathcal{C}_k, j \in \mathcal{C}_l} (x^*_{ij} + x^*_{ji})$. If the maximum flows in this graph $\mathcal{G}^*$ have a value lower than 2, then some of the GSEC (3.22) are violated, and the corresponding constraints are added into the model. To compute the maximum flow in $\mathcal{G}^*$ and detect all violated inequalities, the Gomory-Hu algorithm is applied, with a $\mathcal{O}(K^4)$ time complexity (Gomory & Hu, 1961). By using Gomory-Hu algorithm, we obtain the maximum flow and the corresponding minimum cut between

each pair of distinct vertices in $\mathcal{V}^*$. Thus, all the minimum cuts in $\mathcal{G}^*$ with a value lower than 2 are detected, and the related GSEC (3.22) are added into the model.

3.5.3.2 Separation of the SOP inequalities

π-inequalities

We adapt the procedure used by Balas *et al.* (1995) to separate this family of inequalities. For any vertex i with $\pi(i) \neq \emptyset$ and $y_i^* > 0$, we consider a graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{A}^*)$, such that $\mathcal{V}^* = \mathcal{V} \setminus \{\pi(i) \cup \{\mathcal{C}_{(i)} \setminus \{i\}\}\}$ and $\mathcal{A}^* = \{(i,j) | i,j \in \mathcal{V}^*, x_{ij}^* > 0\}$. We associate with each arc in $\mathcal{A}^*$ a capacity equal to the corresponding x^* values and we compute the maximum flow from vertex i to the depot 0 in $\mathcal{G}^*$. If this flow is less than y_i, then the minimum $(i,0)$ cut identifies a violated π-inequality.

σ-inequalities

To detect violated σ-inequalities we apply a procedure similar to the one described for the π-inequalities except that $\mathcal{V}^* = \mathcal{V} \setminus \{\sigma(i) \cup \{\mathcal{C}_{(i)} \setminus \{i\}\}\}$ and we compute the maximum from the depot 0 to vertex i in the graph $\mathcal{G}^*$. If this flow is lower than y_i, the corresponding minimum cut identifies a violated σ-inequality.

(π, σ)-inequalities

To detect violated (π, σ)-inequalities we consider each pair of vertices i, j such that $i \prec j, \pi(i) \neq \emptyset, \sigma(j) \neq \emptyset$ and $y_i^* + y_j^* - 1 > 0$. We then apply a procedure similar to the one described for the π-inequalities except that $\mathcal{V}^* = \mathcal{V} \setminus \mathcal{W}_{ij}$ where $\mathcal{W}_{ij} = \{0\} \bigcup \pi(i) \bigcup \sigma(j) \bigcup \mathcal{C}_{(i)} \bigcup \mathcal{C}_{(j)} \setminus \{i,j\}$ and we compute the maximum from vertex i to vertex j in the graph $\mathcal{G}^*$. If this flow is lower than $(y_i + y_j - 1)$, the corresponding minimum cut identifies a violated (π, σ)-inequality.

Note that the graph $\mathcal{G}^*$ can be sparsified by deleting: i) all vertices k such that path (i,k,j) is infeasible, i.e., $E_i + SP_{ik} + SP_{kj} > L_j$; ii) all arcs (u,v) such that path (i,u,v,j) is infeasible, e.g., $E_i + SP_{iu} + T_{uv} + SP_{vj} > L_j$.

$\pi_{\mathcal{C}_k}$-inequalities

To separate $\pi_{\mathcal{C}_k}$-inequalities, we consider any cluster $\mathcal{C}_k$ such that $\pi(\mathcal{C}_k) \neq \emptyset, |\mathcal{C}_k| > 1$. $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{A}^*)$ is such that $\mathcal{V}^* = \{s_k\} \cup \mathcal{V} \setminus \pi(\mathcal{C}_k)$ where s_k is an additional vertex and $\mathcal{A}^* = \mathcal{A}_1^* \cup \mathcal{A}_2^* = \{(i,j) | i,j \in \mathcal{V}^*, x_{ij}^* > 0\} \cup \{(s_k, j) | j \in \mathcal{C}_k\}$. We associate with each arc

in $\mathcal{A}_1^*$ a capacity equal to the corresponding x^* value and with each arc in $\mathcal{A}_2^*$ a very large capacity. In the resulting graph $\mathcal{G}^*$, we compute the maximum flow from vertex s_k to the depot 0. If it is strictly less than 1, a violated $\pi_{\mathcal{C}_k}$ inequality is identified.

$\sigma_{\mathcal{C}_k}$-inequalities

These inequalities are detected by adapting the explained procedure to identify violated $\pi_{\mathcal{C}_k}$-inequalities.

$(\pi_{\mathcal{C}_k}, \sigma_{\mathcal{C}_k})$-inequalities

To detect violated $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequalities we consider each pair of clusters $\mathcal{C}_h, \mathcal{C}_k$ such that $\mathcal{C}_h \prec \mathcal{C}_k$, $|\mathcal{C}_h| > 1, |\mathcal{C}_k| > 1$. $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{A}^*)$ is such that $\mathcal{V}^* = \{s_h, s_k\} \cup \mathcal{V} \setminus \{\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k)\}$ where s_h and s_k are two additional vertices and $\mathcal{A}^* = \mathcal{A}_1^* \cup \mathcal{A}_2^* \cup \mathcal{A}_3^* = \{(i,j) | i,j \in \mathcal{V}^*, x_{ij}^* > 0\} \cup \{(s_h, j) | j \in \mathcal{C}_h\} \cup \{(j, s_k) | j \in \mathcal{C}_k\}$. We associate with each arc in $\mathcal{A}_1^*$ a capacity equal to the corresponding x^* value and with each arc in $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$ a very large capacity. In the resulting graph $\mathcal{G}^*$, we compute the maximum flow from vertex s_h to vertex s_k. If it is strictly less than 1, a violated $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequality is identified.

3.5.3.3 Separation of the arc selection inequalities, arc-or-vertex inequalities and clique inequalities

Arc selection inequalities (3.18), arc-or-vertex inequalities (3.19) and their lifted versions (3.20) and (3.21) are polynomial in the size of the input. The clique inequalities (3.35) and their lifted versions (3.36) and (3.37), we consider, are restricted to $\mathcal{S} \subset \mathcal{V}$, $|\mathcal{S} \cap \mathcal{C}_k| \leq 1$ for all $k \in \mathcal{K}$ with $|\mathcal{S}| = 2, 3$. Therefore, whenever a fractional solution is obtained, we scan the entire set of these inequalities to seek for those that are violated.

3.5.3.4 Separation strategy

During the branch-and-cut procedure, at each time a fractional solution is obtained, the separation procedures for GSEC, SOP, clique, arc orientation and arc-or-vertex inequalities is called. GSEC (3.22) are separated using Gomory-Hu algorithm. One call to the algorithm provides all the violated cuts. However, SOP cuts and SOP cuts on clusters require repeated calls to maximum flow algorithm: $\mathcal{O}(N)$ times for π

and σ-inequalities, $\mathcal{O}(N^2)$ times for (π, σ)-inequalities, $\mathcal{O}(K)$ times for $\pi_{\mathcal{C}_k}$ and $\sigma_{\mathcal{C}_k}$-inequalities, and $\mathcal{O}(K^2)$ times for $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequalities.

Solving these maximum flow problems would be time-consuming. Therefore, we introduce a parameter α to control the percentage of SOP inequalities that we choose to separate. For each class of SOP inequalities, we first determine the eligible vertices or clusters (for example for π-inequalities, all the vertices i such that $\pi(i) \neq \emptyset$ and $y_i^* > 0$), and then α percent are randomly chosen to be separated.

In addition, to improve the influence of the cuts added to the relaxed model, they should be significantly violated. Thus, we introduce a parameter ε to control the lowest violation of the cuts we add. For each family of inequalities, after the call to the separation algorithm, only the cuts having a violation of at least ε are added into the model.

Detailed results on the setting of these parameters α and ε are presented in section 3.6.2.

Note that all the separation algorithms are exact. This means that each time a specific separation algorithm is called it provides the corresponding violated cuts if any. However, due to the introduction of the parameter α, the number of calls to the SOP separation is limited and some violated cuts may be missed. Moreover, only the cuts with a least violation ϵ are added. As a consequence, when α and ϵ are introduced, the global separation strategy becomes heuristic even if all the separation procedures are exact.

3.6 Computational experiments

The algorithms were implemented in C++ in Visual Studio environment and uses CPLEX 12.6.3 and the Concert framework. Experiments were performed on a PC Intel(R) Core(TM) i5-6200U CPU 2.20GHz and 64G RAM. The computation time limit (TL) was set to 3600 seconds.

3.6.1 Problem instances

Since no testbed is available for GTSPTW in the literature, we created three groups of instances to test the proposed algorithm.

The first group indicated by *G1* includes 47 instances. *G1* is generated by performing suitable modifications to the existing benchmark for the GTSP proposed by

Karapetyan (2012). The GTSP does not take time into account, so for each arc of the graph, the traveling time is set equal to the traveling cost. The GTSP instances do not necessarily contain a cluster with a unique vertex that could be the depot. Hence, based on a GTSP instance, a depot is added. As coordinates are not always available in GTSP instances, the traveling time from the depot to other vertices is fixed to be 0. The TW of the depot is $[0, T]$, where T initially equals twice the best objective value of the original GTSP instance. Then, the TW $[E_i, L_i]$ for each vertex $i \in \mathcal{V} \setminus \{0\}$ is generated according to the method described by Solomon (1987). The center of the TW, denoted as c_i, is randomly generated from a uniform distribution in the interval $[0, T]$. For the width w_i of the TW, a number r_i is randomly generated from the standard normal distribution, then $w_i = |r_i| \min\{c_i, T - c_i\}$. Thus the TW $[E_i, L_i]$ can be obtained as $E_i = c_i - w_i, L_i = c_i + w_i$. Once the above procedure has been applied, we impose a modification procedure to ensure that a feasible solution exists for the instance created. First, a sequence of clusters is randomly generated. Then, for each cluster, one vertex is randomly selected in order to get a tour. Starting at time 0 from the depot, we compute the service time at each vertex from its previous vertex in this tour. If the service time at vertex i exceeds the upper bound L_i, L_i is then updated with the service time value. At the end of the tour, if the arrival time at the depot is greater than T, then T is updated to the arrival time value. In this way, we ensure that a feasible solution exists for the instance generated. The instance name is obtained by adding TW_ before its original GTSP name, e.g., we create TW_3burma14 based on 3burma14. The number in the middle of the name represents the number of clusters, and the number at the end represents the number of vertices, both excluding the depot.

In GTSP instances, some clusters contain a large number of vertices. This does not correspond to the last mile delivery application. Therefore, we generated a second group indicated by *G2* which includes 40 instances. *G2* is obtained in a similar way to the set *G1*, except that the maximum number of vertices per cluster $N_{\max}$ is fixed. In our experiments, we set $N_{\max} = 5$ which is reasonable in the case of last mile delivery. Based on a GTSP instance, when the number of vertices in a cluster $\mathcal{C}_k$ is greater than $N_{\max}$, we divide the corresponding cluster into M' clusters, where $M' = \lceil |\mathcal{C}_k| / N_{\max} \rceil$. The first $N_{\max}$ vertices in $\mathcal{C}_k$ constitute the first cluster, and so on. The following steps are as described above to create instances in *G1*. The instance name is similar to *G1*, while the number in the middle of the name changes to the number of clusters in the instance excluding the depot, e.g., TW_4burma14 is created based on 3burma14.

The third group with 72 instances, indicated by *G3* is obtained from the instances proposed by Reyes *et al.* (2017) for the VRPRDL. As mentioned above, the TWs in these instances have a particular structure, i.e., the TWs associated with vertices in a same cluster do not overlap. Since the VRPRDL instances are for the multi-vehicle case, we use only part of the instance information. Given one VRPRDL instance, we consider the best corresponding solution, and define an instance for the GTSPTW by the clusters visited along the longest route in the solution. The data about the distances, the traveling times and TWs are not modified.

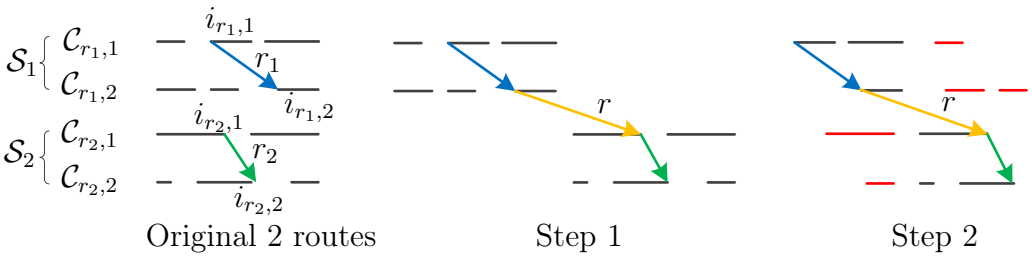


Figure 3.5: Create *G3* instances from 2 routes.

To create larger instances, we combine several routes belonging to a solution of one VRPRDL instance. In this case, TWs need to be modified to obtain a set of customers that can be visited in a single route. Notice that the vertices within the same cluster are ordered based on the earliest visit times. An example is given in Figure 3.5 for the case with two routes. Suppose that $\mathcal{S}_1$ and $\mathcal{S}_2$ are the cluster sets which include all the clusters visited in the routes r_1 and r_2, respectively. We note $(\mathcal{C}_{r_1,1}, \mathcal{C}_{r_1,2}, \ldots, \mathcal{C}_{r_1,|\mathcal{S}_1|})$ the sequence of clusters visited by route r_1, and $(i_{r_1,1}, i_{r_1,2}, \ldots, i_{r_1,|\mathcal{S}_1|})$ the sequence of vertices visited in route r_1. Obviously, $i_{r_1,k} \in \mathcal{C}_{r_1,k}$, $\forall k = 1, \ldots, |\mathcal{S}_1|$. The same notation is used for route r_2. The time horizon of the original VRPRDL instances lasts $\Delta = 12$ hours. We then move forward the TWs of all the vertices belonging to the clusters in $\mathcal{S}_2$ by Δ plus the traveling time from $i_{r_1,|\mathcal{S}_1|}$ the last visited vertex in r_1 to $i_{r_2,1}$ the first visited vertex in r_2. Then for each cluster $\mathcal{C}_{r_1,k}$ in $\mathcal{S}_1$, the TWs of vertices which are before $i_{r_1,k}$ the vertex visited in this cluster by r_1 are moved forward by $\Delta + \delta(\mathcal{C}_{r_1,k})$. $\delta(\mathcal{C}_{r_1,k})$ corresponds to the traveling time from the last vertex of $\mathcal{C}_{r_1,k}$ to the first vertex of $\mathcal{C}_{r_1,k}$, where vertices are ordered based on the earliest visit times. In the same way, for each cluster $\mathcal{C}_{r_2,k}$ in $\mathcal{S}_2$, the TWs of vertices which are after $i_{r_2,k}$ the vertex visited in this cluster by r_2 are moved backward by $\Delta + \delta(\mathcal{C}_{r_2,k})$. In this way, we inherit the property of non-overlapping TWs within a cluster and ensure that there is a feasible

solution $r = [r_1, r_2]$ in the created instance. Similarly, we can use more than two routes of a solution to create instances.

We use sequences *a-b-c-d* to name instances in *G3*, where a and b indicate the number of clusters and vertices in the instance respectively, similar to *G1* and *G2*, excluding the depot. c and d indicate that the instance is created based on the dth VRPRDL instance using its c longest routes in the solution, e.g., 14-41-2-6 means that using the two longest routes in the solution of the sixth VRPRDL instance, we create a *G3* instance with 14 clusters and 41 vertices.

3.6.2 Parameter tuning results

In the separation procedure described in Section 3.5.3.4 we introduced two parameters, α and ε, which respectively control the percentage of SOP inequalities that we randomly choose to separate and the least violation of all the cuts that we add. We limit the impact of α to the SOP inequalities due to their large cardinality. Here we discuss the tuning phase of α and ε.

Based on some preliminary computational results, if we use CPLEX branch-and-bound scheme to solve the instances without any of the proposed valid inequalities, formulation $\mathcal{F}2$ performs slightly better than formulation $\mathcal{F}1$ in terms of resolution time and final optimality gap. Due to the fact that the difference is small, we conduct the tuning on both formulations.

To observe significant differences between results obtained from different parameter settings, we choose instances of comparatively medium and large size in sets *G1* and *G2*. We do not consider instances of *G3* since preliminary results showed that they are the easiest to solve.

We choose 7 instances: 4 in *G1* (TW_20kroA100, TW_25pr124, TW_26ch130, TW_35si175) and 3 in *G2* (TW_28rat99, TW_30eil101, TW_32gr120). We consider values of α in $\{40; 60; 80; 100\}$ and values of ε in $\{0.05; 0.1; 0.15\}$ and perform experiments for all possible pairs (α, ε). The average optimality gaps for the two formulations $\mathcal{F}1$ and $\mathcal{F}2$ are reported in Table 3.1. Computation time is limited to one hour per execution.

Our results in Table 3.1 show that the average optimality gaps obtained from $\mathcal{F}2$ are always smaller than those obtained from $\mathcal{F}1$. This indicates that formulation $\mathcal{F}2$ turns to be superior to formulation $\mathcal{F}1$. It is also clear that the best parameter setting

for $\mathcal{F}2$ is $(\alpha, \epsilon) = (40, 0.15)$ with the smallest average gap of 0.37%. Therefore, we conduct the following experiments using formulation $\mathcal{F}2$ with $(\alpha, \epsilon) = (40, 0.15)$.

Table 3.1: Tuning results on α, ϵ, $\mathcal{F}1$ and $\mathcal{F}2$.

ϵ	α	Average gap $\mathcal{F}1$	Average gap $\mathcal{F}2$
0.05	40	2.02%	0.64%
	60	1.53%	1.15%
	80	2.07%	0.99%
	100	1.91%	1.41%
0.10	40	2.33%	0.58%
	60	2.37%	0.85%
	80	2.51%	1.07%
	100	1.62%	1.32%
0.15	40	1.81%	**0.37%**
	60	1.70%	0.75%
	80	2.02%	1.19%
	100	1.62%	0.76%

3.6.3 Effectiveness of families of valid inequalities

In order to assess the effectiveness of each family of the proposed valid inequalities, i.e., to know which are important and which provide marginal improvements, we compute the lower bounds at the root node (after applying the preprocessing steps) when only one of the families of inequalities is part of the branch-and-cut algorithm, as well as when only one family is excluded. We conduct the computations on instances in *G1* with at least 7 clusters. The average results are shown in Table 3.2. We divide the results into two parts, i.e., *with one family* and *without one family*.

Columns *with one family* correspond to the results obtained when applying a single family of inequalities. There are eight configurations. Row *LP* reports results of the linear relaxation. Row *CPLEX* shows the results when activating the automatic cut generation of CPLEX (but not the generation of user cuts). The rows *Arc selection*, *Arc-or-Vertex*, *Clique*, *GSEC* and *SOP* report the results obtained by including the automatic cut generation of CPLEX and the corresponding family of inequalities. Finally, row *Full* reports the results obtained when applying the automatic cut generation of CPLEX and the separation algorithms for all families of valid inequalites.

Columns *without one family* correspond to the results obtained when one of the families of inequalities is excluded. There are five configurations. Rows *Arc selection*,

Table 3.2: Average lower bound results at the root node obtained with/without one family of inequalities.

Configuration	**with** one family			**without** one family		
	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts
LP	**0**	**0.18**				
CPLEX	13.53	2.44				
Arc selection	30.13	3.29	85.49	-0.60	45.69	1096.62
Arc-or-vertex	13.71	3.23	163.92	-0.21	32.56	828.24
Clique	13.65	3.34	37.30	-0.13	33.76	942.22
GSEC	18.38	2.49	21.97	-0.02	37.04	1055.62
SOP	53.77	49.62	1029.84	-10.40	3.08	318.76
Full	55.12	33.50	990.86	**0**	**33.50**	**990.86**

Arc-or-Vertex, *Clique*, *GSEC* and *SOP* report the results obtained by including the automatic cut generation of CPLEX and generating violated inequalities without the family indicated by the row name.

Three average statistics are reported in Table 3.2. Column *Impr(%)* of *with one family* represents the average improvement in percentage of lower bounds obtained using one configuration compared with the bounds obtained by solving linear relaxations. The value *Impr(%)* for a single instance is calculated by $Impr\% = (LB_{config} - LB)/LB * 100\%$, where LB_{config} is the lower bound obtained at the root node under the corresponding configuration and LB the bound obtained by solving the linear relaxation of the problem. Column *Impr(%)* of *without one family* represents the average improvement of lower bounds obtained under the configuration compared with those obtained by separating all families of inequalities. The value *Impr(%)* for a single instance is calculated as $Impr\% = (LB_{config} - LB_{full})/LB_{full} * 100\%$ where LB_{full} denotes the lower bound obtained at the root node when all families of valid inequalities are considered. Columns $time(s)$ represent the average computation times in seconds. Columns *nbCuts* indicate the average numbers of violated cuts added under a configuration. Note that we marked the comparison baselines in bold, i.e., results of *LP* and *Full* for *with one family* and *without one family* respectively.

From the average results of *with one family*, we can order the five families of valid inequalities according to their effectiveness as follows: SOP > Arc selection > GSEC > Arc-or-Vertex > Clique. The largest improvement is obtained with the SOP, followed by Arc Selection. Compared with the other families of valid inequalities, the separation of the SOP requires more computational time and identifies more violated

cuts. From the results of column *Full*, it can be seen that separating all families of valid inequalities consumes less time than separating only the SOP, meanwhile this configuration improves the lower bound. When all the valid inequalities are separated, the lower bound at the root node improves by 55.12% on average, compared with the bound obtained by solving the linear relaxation.

From the average results of *without one family*, we can order the valid inequalities according to their effectiveness as follows: SOP > Arc selection > Arc-or-Vertex > Clique > GSEC. It can be seen that without either family, the lower bound quality decreases. The most significant decrease in both the lower bound quality and computation time is obtained when the SOP inequalities are excluded.

In general, the SOP inequalities are crucial for the efficiency of the branch-and-cut algorithm, while other inequalities bring smaller improvements. However, the separation of all families of valid inequalities allows to obtain overall good performances in terms of lower bound and computation time.

3.6.4 Computational results

We now present the results obtained with the branch-and-cut algorithm presented in this paper. The algorithm was tested on the 158 instances in *G1*, *G2* and *G3* and, due to the results presented in Section 3.6.2, experiments were carried out with formulation $\mathcal{F}2$. Table 3.3 provides the column headings used in the following tables. Detailed results on *G1*, *G2* and *G3* are presented in Table 3.4, 3.5 and 3.6 respectively.

Table 3.3: Column headings.

Column heading	Description
Instance	name of the problem instance
Obj	value of the best solution obtained
initS	value of initial heuristic solution
rLB	lower bound at the root node
fLB	lower bound at the termination/at the time limit
rGAP	gap at the root node: rGAP $= \frac{\text{rUB}-\text{rLB}}{\text{rUB}}$, rUB is the upper bound at the root node
fGAP	gap at the termination/at the time limit: fGAP $= \frac{\text{Obj}-\text{fLB}}{\text{Obj}}$
nbNode	number of nodes in the branch-and-cut tree
nbCuts	number of generated cuts
sep-time	time spent in separation procedure in seconds
time	total computation time in seconds (equals to 1 hour if optimality is not achieved)

As mentioned in Section 3.5.2, the labeling algorithm can be executed on all sequences with less than seven clusters within reasonable computation times. In this case,

instead of searching for an initial solution, we just enumerate all feasible sequences to get an optimal solution without calling the branch-and-cut algorithm. For this reason, we report only the objective values and computation times for the first ten lines of Table 3.4, the first four lines of Table 3.5 and the first twelve lines of Table 3.6. Rows *Average* in three tables report the average results for instances with more than 6 clusters.

In Table 3.4, we report results on instances in *G1*. Our results indicate that our algorithm is able to solve most instances up to 30 clusters and 150 vertices within one hour. However, one instance with 26 clusters (TW_26bier127), one instance with 28 clusters (TW_28pr136) and two instances with 30 (TW_30ch150, TW_30kroB150) clusters remain unsolved. Moreover, all instances with up to 24 clusters and 124 vertices are systematically solved in less than three minutes. For all the instances ($K \geq 7$) solved to optimality, we calculate the average gap between their initial solution obtained from the heuristic $initS$ and the final optimal solution Obj using $(initS - Obj)/Obj * 100\%$, which equals to 0.78%. This proves the efficiency of the heuristic.

When we consider instances in *G2* (see Table 3.5), we can observe that the largest solved instance within the given time frame involves 38 clusters and 150 vertices. Instances with up to 30 clusters and 101 vertices are systematically solved to optimality in one hour of computation time. Initial heuristic is also very efficient on instance set *G2*. For all the instances ($K \geq 7$) solved to optimality, the average gap between their initial solution obtained from the heuristic $initS$ and the final optimal solution Obj equals to 0.23%.

Table 3.6 reports results on instances in *G3* (see Table 3.6). All the instances are solved to optimality in very short computation times. Most instances are solved at the root node. The largest instance with 32 clusters and 125 vertices is solved to optimality in less than one minute. For most of the instances, the initial solutions obtained from the heuristic turn out to be the optimal solutions. From the results, we can see that instances in *G3* are easier to solve than instances in *G1* and *G2*. One of the possible reasons may be due to the special non-overlapping TW structure for vertices in the same cluster.

The computational results on the three groups of instances show that the proposed branch-and-cut algorithm is very efficient and is able to solve instances with 30 clusters. This size is near to the number of deliveries that a courier might make during a day in

an urban context. Moreover, the initial solutions provided by the heuristic are of high quality with respect to the best solutions obtained.

Table 3.4: Results on *G1* ($\mathcal{F}2$).

Instance	Obj	initS	rGAP(%)	rLB	fGAP(%)	fLB	nbNodes	nbCuts	sep-time(s)	time(s)
TW_3burma14	908	908								0.01
TW_4br17	19	19								0.01
TW_4gr17	962	962								0.01
TW_4ulysses16	2392	2392								0.01
TW_5gr21	1165	1165								0.01
TW_5gr24	263	263								0.01
TW_5ulysses22	3287	3287								0.01
TW_6bayg29	476	476								0.02
TW_6bays29	628	628								0.02
TW_6fri26	354	354								0.02
TW_7ftv33	416	416	0.00	416.0	0.00	416.0	0	53	0.02	1.49
TW_8ftv36	538	554	6.10	520.2	0.00	538.0	11	162	0.28	3.09
TW_8ftv38	384	410	0.00	384.0	0.00	384.0	0	104	0.03	2.20
TW_9dantzig42	322	322	5.55	304.1	0.00	322.0	105	575	2.01	7.24
TW_10att48	4113	4113	0.00	4113.0	0.00	4113.0	0	166	0.06	5.93
TW_10gr48	1437	1515	2.49	1429.5	0.00	1437.0	3	190	0.31	7.13
TW_10hk48	5268	5268	4.10	5052.1	0.00	5268.0	52	484	1.24	7.27
TW_11berlin52	3632	3632	9.69	3280.1	0.00	3632.0	346	891	5.84	14.16
TW_11eil51	151	157	0.00	151.0	0.00	151.0	0	262	0.61	6.52
TW_12brazil58	13503	13503	2.79	13126.5	0.00	13503.0	69	607	1.92	10.33
TW_14st70	289	289	5.67	272.6	0.00	289.0	1137	1852	44.89	75.37
TW_16eil76	205	205	3.78	197.3	0.00	205.0	70	2015	6.05	28.39
TW_16pr76	57164	57164	0.96	56614.5	0.00	57164.0	9	724	2.77	22.11
TW_20gr96	33128	33128	3.71	31898.0	0.00	33128.0	86	1515	24.48	78.34
TW_20kroA100	10209	10209	7.95	9397.7	0.00	10209.0	139	4132	66.72	158.07
TW_20kroB100	9862	9975	3.12	9598.0	0.00	9862.0	111	3724	37.13	103.99
TW_20kroC100	9728	9728	3.13	9423.7	0.00	9728.0	117	3031	29.70	81.46
TW_20kroD100	9210	9210	3.29	8906.5	0.00	9210.0	20	1343	18.32	66.06
TW_20kroE100	9514	9514	3.18	9211.3	0.00	9514.0	130	3737	21.77	71.42
TW_20rat99	478	478	6.60	446.4	0.00	478.0	138	2734	33.62	91.42
TW_20rd100	3446	3505	3.99	3365.3	0.00	3446.0	316	2802	37.87	99.85
TW_21eil101	247	247	5.43	233.6	0.00	247.0	322	3654	51.34	130.44
TW_21lin105	7582	7582	2.27	7410.2	0.00	7582.0	53	3484	22.01	79.92
TW_22pr107	21352	21481	0.76	21190.5	0.00	21352.0	6	1540	10.80	63.11
TW_24gr120	2811	2825	4.95	2685.0	0.00	2811.0	100	3584	55.29	174.82
TW_25pr124	36520	36520	6.90	34001.2	0.00	36520.8	16115	11990	1568.63	3286.21
TW_26bier127	86761	86761	16.88	72115.0	15.30	73485.5	4608	12752	1374.56	3600.00
TW_26ch130	2891	2896	5.17	2746.2	0.00	2891.0	5027	10137	1230.36	2346.46
TW_28gr137	34123	34201	4.59	32631.2	0.00	34123.0	2223	6696	347.50	712.83
TW_28pr136	45998	45998	7.49	42554.8	2.99	44622.5	2823	17470	1925.40	3600.00
TW_29pr144	43639	43639	1.12	43149.6	0.00	43639.0	107	4185	135.03	301.12
TW_30ch150	2895	2895	6.94	2694.1	2.02	2836.5	4838	12829	1957.07	3600.00
TW_30kroA150	11475	11475	4.35	10975.7	0.00	11475.0	251	6627	204.27	476.97
TW_30kroB150	12802	12869	6.93	11977.0	3.94	12297.7	2575	13882	2010.46	3600.00
TW_31pr152	55128	55128	17.32	45580.4	16.22	46185.5	4170	16404	1361.59	3600.00
TW_32u159	22846	22846	7.05	21235.9	2.57	22259.8	1682	11944	1787.29	3600.00
TW_35si175	5521	5561	2.38	5428.6	0.92	5470.2	1149	11657	1892.65	3600.00
Average	15286.2	15303.2	4.77	14181.5	1.19	14616.1	1321.8	4863.2	439.73	911.18

Table 3.5: Results on *G2* ($\mathcal{F}2$).

Instance	Obj	initS	rGAP(%)	rLB	fGAP(%)	fLB	nbNodes	nbCuts	sep-time(s)	time(s)
TW_4burma14	957	957								0.01
TW_5br17	19	19								0.01
TW_5gr17	766	766								0.01
TW_6ulysses16	2521	2521								0.01
TW_7gr21	1475	1475	0.00	1475.0	0.00	1475.0	0	41	0.01	0.87
TW_7gr24	365	365	0.00	365.0	0.00	365.0	0	25	0.01	0.99
TW_8ulysses22	3461	3461	0.00	3461.0	0.00	3461.0	0	63	0.01	1.24
TW_8bayg29	515	515	0.00	515.0	0.00	515.0	0	74	0.02	1.36
TW_8bays29	595	595	0.00	595.0	0.00	595.0	0	32	0.01	1.73
TW_9fri26	498	498	0.00	498.0	0.00	498.0	0	49	0.00	1.57
TW_10ftv33	467	467	0.00	467.0	0.00	467.0	0	121	0.16	2.97
TW_10ftv36	500	500	0.00	500.0	0.00	500.0	0	126	0.25	3.36
TW_10ftv38	469	469	8.60	428.7	0.00	469.0	51	439	1.02	4.69
TW_12dantzig42	364	364	0.00	364.0	0.00	364.0	0	214	0.61	5.48
TW_13gr48	2010	2010	1.78	1974.1	0.00	2010.0	8	615	0.39	5.64
TW_14att48	5415	5415	6.98	5037.2	0.00	5415.0	238	946	8.28	20.22
TW_14hk48	5989	5989	6.28	5612.7	0.00	5989.0	191	883	5.16	14.92
TW_15eil51	172	172	3.20	166.5	0.00	172.0	5	354	1.06	10.95
TW_16berlin52	3821	3821	3.79	3676.2	0.00	3821.0	307	947	6.10	19.28
TW_17brazil58	12665	12665	3.70	12196.4	0.00	12665.0	107	1051	9.49	27.12
TW_20st70	329	329	5.13	312.1	0.00	329.0	113	1299	10.54	36.50
TW_21eil76	245	245	9.59	221.5	0.00	245.0	6299	3081	320.90	486.03
TW_22pr76	71098	71098	6.61	66401.1	0.00	71098.0	132	2188	33.64	79.20
TW_24kroC100	9932	9932	3.39	9595.5	0.00	9932.0	22	1754	20.57	89.79
TW_26kroA100	10515	10541	3.58	10139.0	0.00	10515.0	64	2555	20.97	83.92
TW_28rat99	618	628	10.91	559.5	0.00	618.0	4043	8664	955.48	1815.03
TW_28kroB100	11937	11976	5.64	11301.1	0.00	11937.0	485	4695	75.53	163.44
TW_28kroD100	10720	10720	8.85	9771.0	0.00	10720.0	162	3716	112.92	251.70
TW_29gr96	35064	35064	7.86	32309.3	0.00	35064.0	2178	6008	427.12	718.39
TW_29kroE100	11275	11275	5.59	10645.1	0.00	11275.0	11531	5605	677.56	1113.05
TW_29rd100	4380	4380	10.12	3936.8	0.00	4380.0	5586	8227	898.58	1854.56
TW_30eil101	292	292	6.30	273.6	0.00	292.0	810	3946	205.87	350.24
TW_32lin105	9852	9965	10.17	8951.6	2.67	9588.9	8515	9258	1848.66	3600.00
TW_32gr120	3307	3307	9.20	3002.9	1.66	3252.0	5938	12849	1779.74	3600.00
TW_32pr124	37655	37655	7.14	34967.9	2.94	36549.8	5110	13172	1974.32	3600.00
TW_34pr107	34349	34349	19.27	27728.5	14.20	29471.8	7063	12334	1382.09	3600.00
TW_36ch130	3309	3345	3.88	3215.2	0.00	3309.0	3242	5399	560.62	922.58
TW_37pr144	50912	51377	8.71	46901.5	7.29	47200.5	765	13594	1527.02	3600.00
TW_38ch150	3175	3288	7.38	3016.5	0.00	3175.0	2890	11307	1533.51	2807.53
TW_39gr137	41177	41177	7.70	38006.0	5.28	39004.0	3209	9428	2556.84	3600.03
Average	10803.4	10825.7	5.32	9960.8	0.95	10464.9	1918.4	4029.4	470.97	902.62

Table 3.6: Results on *G3* ($\mathcal{F}2$).

Instance	Obj	initS	rGAP(%)	rLB	fGAP(%)	fLB	nbNodes	nbCuts	sep-time(s)	time(s)
5-10-1-3	326	326								0.01
5-13-1-1	329	329								0.01
5-14-1-2	186	186								0.01
5-25-1-15	231	231								0.01
6-15-1-9	201	201								0.01
6-16-1-5	292	292								0.01
6-19-1-22	301	301								0.01
6-22-1-4	344	344								0.01
6-24-1-0	227	227								0.01
6-25-1-7	312	312								0.01
6-25-1-17	315	315								0.01
6-29-1-11	283	283								0.01
7-21-1-18	271	271	0.00	271.0	0.00	271.0	0	0	0.00	0.49
7-26-1-27	299	299	0.00	299.0	0.00	299.0	0	0	0.00	0.57
7-27-1-13	267	267	0.00	267.0	0.00	267.0	0	0	0.00	0.68
7-27-1-14	312	312	0.00	312.0	0.00	312.0	0	17	0.01	0.63
7-30-1-29	354	354	0.00	354.0	0.00	354.0	0	44	0.01	0.55
7-31-1-20	245	245	0.00	245.0	0.00	245.0	0	0	0.00	0.65
8-14-1-19	318	318	0.00	318.0	0.00	318.0	0	18	0.01	0.49
8-24-1-10	281	281	0.00	281.0	0.00	281.0	0	103	0.02	0.83
8-24-1-12	273	273	0.00	273.0	0.00	273.0	0	37	0.03	0.91
8-27-1-28	306	306	0.00	306.0	0.00	306.0	0	0	0.00	0.73
8-28-1-8	213	213	0.00	213.0	0.00	213.0	0	19	0.00	0.70
8-29-1-38	229	229	0.00	229.0	0.00	229.0	0	41	0.00	0.84
8-30-1-25	272	272	0.00	272.0	0.00	272.0	0	24	0.00	0.80
8-36-1-23	278	279	0.00	278.0	0.00	278.0	0	73	0.02	0.87
9-26-1-35	347	347	0.00	347.0	0.00	347.0	0	59	0.02	0.77
9-27-1-21	341	341	0.00	341.0	0.00	341.0	0	141	0.02	0.64
9-34-1-16	333	333	0.00	333.0	0.00	333.0	0	36	0.02	1.00
9-34-1-26	232	232	0.00	232.0	0.00	232.0	0	109	0.02	1.22
10-29-1-6	281	281	0.00	281.0	0.00	281.0	0	34	0.02	1.02
10-32-1-31	350	350	0.00	350.0	0.00	350.0	0	270	0.05	1.28
10-40-1-36	324	324	0.00	324.0	0.00	324.0	0	75	0.03	1.48
10-41-1-30	196	196	0.00	196.0	0.00	196.0	0	156	0.02	1.75
10-42-1-24	258	258	0.00	258.0	0.00	258.0	0	170	0.03	2.02
11-27-1-37	333	333	0.00	333.0	0.00	333.0	0	48	0.02	1.22
11-33-1-39	312	312	0.00	312.0	0.00	312.0	0	118	0.03	1.62
11-43-1-32	347	347	0.00	347.0	0.00	347.0	0	195	0.02	1.31
12-45-1-34	260	260	0.00	260.0	0.00	260.0	0	81	0.03	2.20
13-41-1-33	347	347	0.00	347.0	0.00	347.0	0	327	0.10	2.34
14-34-2-19	478	478	0.00	478.0	0.00	478.0	0	127	0.06	2.08
14-41-2-6	514	514	0.00	514.0	0.00	514.0	0	260	0.02	2.67
14-44-2-27	575	575	0.00	575.0	0.00	575.0	0	258	0.11	3.41
14-44-2-28	500	500	0.00	500.0	0.00	500.0	0	76	0.03	3.48
14-52-2-12	622	622	0.00	622.0	0.00	622.0	0	199	0.08	2.80
15-56-2-23	531	531	0.00	531.0	0.00	531.0	0	312	0.05	4.47
16-51-2-38	429	431	0.00	429.0	0.00	429.0	0	384	0.16	5.45
16-60-2-25	589	589	0.00	589.0	0.00	589.0	0	762	0.06	3.58
16-60-2-26	498	498	0.00	498.0	0.00	498.0	0	324	0.06	4.94
16-69-2-24	415	415	0.00	415.0	0.00	415.0	0	1038	0.06	5.53

Instance	Obj	initS	rGAP(%)	rLB	fGAP(%)	fLB	nbNodes	nbCuts	sep-time(s)	time(s)
17-48-2-35	606	606	9.28	549.7	0.00	606.0	13	961	0.95	5.05
17-50-2-21	563	563	0.00	563.0	0.00	563.0	0	244	0.06	2.64
19-61-2-31	564	564	6.85	525.4	0.00	564.0	19	1451	3.33	12.30
19-72-2-30	510	510	0.00	510.0	0.00	510.0	0	279	0.08	8.73
20-60-2-37	600	600	0.00	600.0	0.00	600.0	0	1235	0.36	5.56
20-82-2-36	643	643	0.00	643.0	0.00	643.0	0	677	0.14	7.06
21-71-2-39	554	554	0.00	554.0	0.00	554.0	0	497	0.41	9.47
21-78-2-32	662	662	0.00	662.0	0.00	662.0	0	844	0.28	10.64
23-83-2-34	448	448	0.00	448.0	0.00	448.0	0	448	0.23	12.70
23-91-2-33	532	532	0.00	532.0	0.00	532.0	0	635	0.34	12.50
24-76-3-21	835	835	0.00	835.0	0.00	835.0	0	1131	1.39	10.72
24-78-3-38	677	677	1.61	666.1	0.00	677.0	8	4912	1.87	12.91
24-86-3-25	850	850	0.00	850.0	0.00	850.0	0	630	0.41	15.39
25-76-3-35	830	830	0.00	830.0	0.00	830.0	0	1621	1.26	9.53
28-99-3-31	844	850	2.46	823.2	0.00	844.0	7	5475	11.45	34.00
28-110-3-30	720	720	0.00	720.0	0.00	720.0	0	1734	1.17	28.70
29-95-3-37	850	850	3.27	822.2	0.00	850.0	5	6896	6.29	25.50
29-95-3-39	850	850	0.00	850.0	0.00	850.0	0	1654	0.75	20.95
30-114-3-36	929	929	0.00	929.0	0.00	929.0	0	1758	0.49	24.01
31-118-3-32	921	921	0.00	921.0	0.00	921.0	0	3100	0.72	22.05
31-128-3-33	856	856	0.00	856.0	0.00	856.0	0	8642	1.38	32.29
32-125-3-34	721	721	0.00	721.0	0.00	721.0	0	2913	0.72	40.51
Average	447.8	447.9	0.39	479.0	0.00	481.6	0.9	894.5	0.59	5.99

The instances in *G1* and *G2* are derived from the same GTSP instances. Instances in *G2* have the maximum number of vertices per cluster set to $N_{\max} = 5$. This feature does not seem to make instance resolution in *G2* any easier than in *G1*. At first glance, one could say that larger instances can be solved in *G2* than in *G1* (38 clusters versus 30 clusters). On the other hand, a more in-depth analysis shows that instances with the same number of vertices are solved faster when they belong to *G1* rather than to *G2*. This can be seen by comparing the average results of 26 instances in *G2* with no less than 48 vertices and their corresponding instances in *G1*. The average computation time is 447.03 seconds for instances in *G1* while it increases to 1248.85 seconds for instances in *G2*. Note that if an instance is not solved to optimality, we consider the computation time as one hour. Therefore, it seems that instances with the same number of vertices but a larger number of clusters are more difficult to solve.

3.7 Conclusions

In this paper, we have presented two formulations $\mathcal{F}1$ and $\mathcal{F}2$ for the Generalized Traveling Salesman Problem with Time Windows, a new generalization of the classical TSPTW and GTSP. We proposed several families of valid inequalities, which contain

polynomial or exponential numbers of constraints. They were incorporated in a branch-and-cut framework through dedicated separation procedures. A high quality initial solution was constructed based on a heuristic and used as a warm start in the branch-and-cut algorithm. We tested the algorithm on three groups of instances with different characteristics. The results clearly demonstrate the efficiency of the proposed branch-and-cut algorithm and the quality of formulation $\mathcal{F}2$. The proposed branch-and-cut algorithm based on formulation $\mathcal{F}2$ can solve instances around 30 clusters within one hour of computation time.

3.8 Acknowledgment

This work is partially supported by the CSC (China Scholarship Council) and by the ELSAT 2020 project. This support is gratefully acknowledged. Thanks are also due to the referees for their valuable comments.

Chapter 4

A column generation based heuristic for the GVRPTW

Contents

This chapter corresponds to the paper "A column generation based heuristic for the generalized vehicle routing problem with time windows", to be submitted shortly. This is a collaborative project with Professor Daniele Vigo from University of Bologna, Italy.

Abstract: The generalized vehicle routing problem with time windows (GVRPTW) is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where the vertex set $\mathcal{V}$ is partitioned into clusters. One cluster contains only the depot, where is located a homogeneous fleet of vehicles, each with a limited capacity. The other clusters represent customers. Each cluster is associated with a demand. Inside a cluster, the vertices represent the possible locations of the customer. Each vertex is associated with a time window, during which the visit must take place if the vertex is visited. The objective is to find a set of routes such that the total traveling cost is minimized, exactly one vertex per cluster is visited, and all the capacity and time constraints are respected. This paper presents a compact integer linear programming formulation of the GVRPTW. An extended formulation based on a set covering model is used to provide a column generation based heuristic to solve the GVRPTW. The proposed solving method combines several components including a construction heuristic, a route optimization procedure, local search operators and the generation of negative reduced cost routes. Experimental results on benchmark instances show that the proposed algorithm is efficient and high-quality solutions for instances with up to 120 clusters are obtained within short computation times.

4.1 Introduction

Nowadays, e-commerce is used on a daily basis and allows customers to purchase online whenever and whatever they like. Customers are no longer restricted to go to a specific store and to respect the opening hours. At the end of 2018, global e-commerce sales reached approximately $2.8 trillion and are estimated to hit $4.5 trillion in 2021 (Wardini, 2018). This growing e-commerce poses a huge challenge for the last mile delivery since the ordered items need to be delivered to individual customers.

Currently, there exist several last mile delivery services to deliver packages to customers. The most common delivery option is home/workplace delivery. Customers

wait at home/workplace to get their packages. Besides, the delivery can be made to pick-up points such as dedicated lockers or stores. In this case, customers can retrieve their packages after delivery has been accomplished. To give an idea, there are more than 2800 lockers located across the US (Holsenbeck, 2018). When customers shop online, they can choose a nearby locker as a delivery location. This reduces the fragmentation of the deliveries in the last mile, thereby helping to reduce the congestion and environmental pollution caused by urban freight trips (Morganti *et al.*, 2014), as well as reducing routing costs. In recent years, a new concept called *trunk/in-car delivery*, has been proposed. Here, customers' packages can be delivered to the trunks of cars. Volvo launched its world-first in-car delivery service in Sweden in 2016 (Kirsten, 2016). In April 2018, Amazon launched the in-car service in partnership with two major automakers General Motors and Volvo. This service is available in 37 cities in the US (Hawkins, 2018). Trunk delivery is different from home/workplace delivery and pick-up points delivery since the car moves and can be in different locations during different periods of time, e.g., parked at the workplace during the morning and at the commercial center during the afternoon. As a consequence, synchronization between the car and the courier is required to accomplish the delivery.

All these delivery services can be combined, and instead of selecting one delivery location during the online purchase, a customer can propose a set of delivery locations with the associated time constraints. To deliver a package to a specific customer, the courier only needs to choose one of the locations provided by the customer.

In this paper, we aim to develop an efficient solution method for the routing problem in the context of last mile delivery including multiple delivery services: home/workplace, pick-up points and car trunk. The last mile delivery with multiple delivery options allows customers to choose several locations to receive their packages. This provides customers more flexibility considering their own convenience. In addition, it could increase the rate of successful first-time deliveries and decrease delivery costs.

In Figure 4.1, we provide an example of the routing problem considered in this context. Six customers are represented with their associated locations grouped into a dotted circle, representing the different *clusters*. Every possible delivery location is associated with a time window (TW) during which delivery should occur. In the case of home or trunk delivery, the TW represents the period when the customer or the customer's car is present at that location. In the case of a pick-up point, the TW represents the period during which the courier can deliver the package before the

customer arrives at the location and picks it up. The problem is to jointly determine the exact location visited for each customer, and for each vehicle, the customers delivered and the sequence of visits while satisfying TW and capacity restrictions. The feasible solution given in the example in Figure 4.1 involves two vehicles, each of them serving three customers.

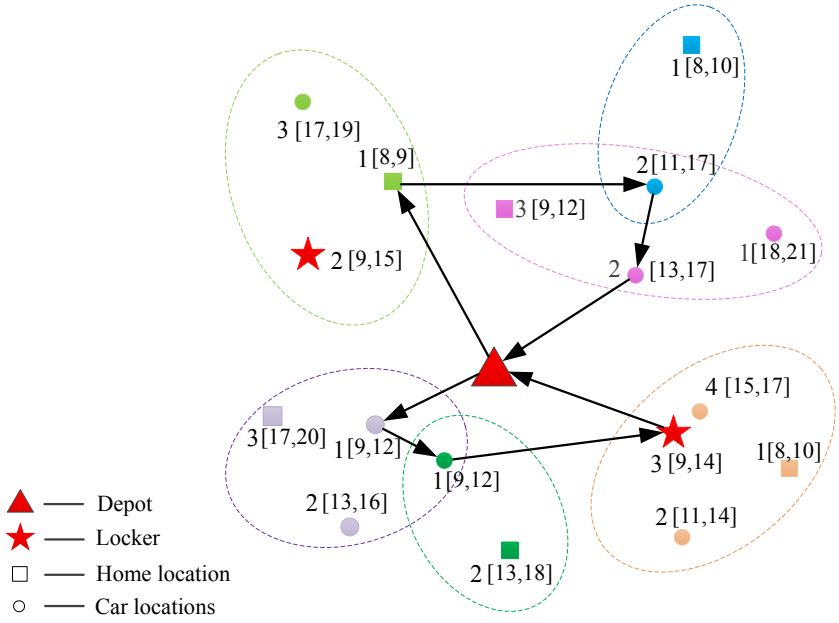


Figure 4.1: An example of the routing problem in the context of last mile delivery with multiple delivery services.

The underlying problem of this practical application is the Generalized Vehicle Routing Problem with Time Windows (GVRPTW), where clusters represent possible delivery locations associated with a customer. When TWs are not considered, the GVRPTW reduces to the Generalized Vehicle Routing Problem (GVRP)(Bektaş *et al.*, 2011). The special case of the GVRPTW where TWs of the locations associated with the same customer do not overlap is called the Vehicle Routing Problem with Roaming Delivery locations (VRPRDL)(Reyes *et al.*, 2017).

This paper introduces the generalized vehicle routing problem with time windows that to the best of our knowledge has not been studied before. Based on a set covering formulation, we propose an efficient column generation based heuristic to solve the GVRPTW. This heuristic solving method relies on a construction heuristic, a route optimization procedure, local search operators, and a heuristic procedure to provide negative reduced cost routes. All the procedures mentioned above take into account the

main characteristic of the GVRPTW: given a customer in a route, we can choose the location to visit. The proposed solution method is tested on different sets of instances from the literature. Results proved the efficiency of the solving method.

The remainder of this paper is organized as follows. A formal description of the problem and an integer programming formulation for the GVRPTW are provided in Section 4.2. Section 4.3 presents the related literature. The description of the heuristic is given in Section 4.4, including the set covering model, the construction heuristic, the route optimization, the local search operators, and the negative reduced cost route generation. Section 4.5 provides details about the experiments and reports the computational results. Finally, conclusions are drawn in Section 4.6.

4.2 Problem description and formulations

The GVRPTW can be formally defined as follows. Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, the set of vertices $\mathcal{V} = \{0, 1, ..., N\}$ is partitioned into $\mathcal{C}_0 = \{0\}, \mathcal{C}_1, ..., \mathcal{C}_K$ clusters, where $\mathcal{K} = \{0, 1, ..., K\}$ denotes the cluster index set. Hence, we have $\bigcup_{k=0}^{K} \mathcal{C}_k = \mathcal{V}$ and $\mathcal{C}_k \bigcap \mathcal{C}_{k'} = \emptyset$, $\forall k, k' \in \mathcal{K}$. Cluster $\mathcal{C}_0$ contains only the depot 0 where a fleet of M homogeneous vehicles is located. Each vehicle has a capacity Q. Cluster $\mathcal{C}_k, k \in \mathcal{K} \setminus \{0\}$ represents the set of alternative locations in which customer k can be delivered. Moreover, each customer is associated with a demand $Q_k, k > 0$. We suppose that the demand at the depot Q_0 is 0. Each vertex is associated with a time window (TW) $[E_i, L_i], i \in \mathcal{V}$ with $[E_0, L_0] = [0, T]$ representing the overall time horizon. A visit can only be made to a vertex during its TW, and an early arrival leads to a waiting time while a late arrival causes infeasibility. Without loss of generality we suppose that the loading and service times are equal to zero. The arc set contains arcs that link vertices belonging to different clusters, that is, $\mathcal{A} = \{(i, j) | i \in \mathcal{C}_k, j \in \mathcal{C}_l, k \neq l, k, l \in \mathcal{K}\}$. Each arc $(i, j) \in \mathcal{A}$ is associated with a traveling cost C_{ij} and a traveling time T_{ij}.

The GVRPTW consists of finding a set of M vehicle routes on $\mathcal{G}$ such that the traveling cost is minimized and: (i) every route starts and ends at the depot; (ii) exactly one vertex from each cluster is visited by a single vehicle; (iii) the sum of the customer demands served by the same vehicle does not exceed Q; (iv) the service at vertex i starts during its TW $[E_i, L_i]$, and (v) every vehicle leaves and returns to the depot during $[0, T]$.

4.2.1 A compact formulation for the GVRPTW

The GVRPTW can be described with a mixed integer linear programming formulation that uses four sets of variables:

- $x_{ij} \in \{0,1\}$: equal to one if and only if arc $(i,j) \in \mathcal{A}$ belongs to the solution;
- $y_i \in \{0,1\}$: equal to one if and only if vertex $i \in \mathcal{V}$ is selected to be visited in the solution;
- $t_i \in \mathbb{R}_+$: the service time at vertex $i \in \mathcal{V}$;
- $q_i \in \mathbb{R}_+$: the cumulative quantity delivered by the vehicle that serves vertex $i \in \mathcal{V}$ when it is leaving it.

Note that, for the depot, t_0 corresponds to the departure time from the depot.

Before presenting the mixed integer linear programming formulation for the GVRPTW, let us introduce the following notation. $Id(i)$ denotes the index of the cluster that contains vertex i, thus $i \in \mathcal{C}_k \Leftrightarrow Id(i) = k$. Given a vertex $i \in \mathcal{V}$, $\delta^+(i) = \{(i,j) \in \mathcal{A} | j \in \mathcal{V} \setminus \{i\}\}$ is the set of arcs leaving vertex i, and $\delta^-(i) = \{(j,i) \in \mathcal{A} | j \in \mathcal{V} \setminus \{i\}\}$ is the set of arcs entering vertex i.

The mathematical programming formulation is as follows:

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{C}_k} y_i = 1 \qquad \forall k \in \mathcal{K}, \tag{4.2}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{4.3}$$

$$\sum_{(0,j)\in\delta^+(0)} x_{0j} \leq M \tag{4.4}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{ij} = \sum_{(j,i)\in\delta^-(i)} x_{ji} \qquad \forall i \in \mathcal{V}, \tag{4.5}$$

$$t_i - t_j + T_{ij} x_{ij} \leq L_i y_i - E_j y_j - (L_i - E_j) x_{ij} \qquad \forall (i,j) \in \mathcal{A}, j \neq 0, \tag{4.6}$$

$$E_i y_i \leq t_i \leq L_i y_i \qquad \forall i \in \mathcal{V}, \tag{4.7}$$

$$t_i + T_{i0} x_{i0} \leq L_0 \qquad \forall i \in \mathcal{V} \setminus \{0\}, \tag{4.8}$$

$$q_i - q_j + Q_{Id(j)} x_{ij} \leq Q(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A}, j \neq 0, \tag{4.9}$$

$$Q_{Id(i)} \leq q_i \leq Q \qquad \forall i \in \mathcal{V}, \tag{4.10}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{V}, \tag{4.11}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}. \tag{4.12}$$

The objective function (4.1) is to minimize the total cost. Constraints (4.2) ensure that each cluster is visited at one of its vertices exactly. Constraints (4.3) impose that one arc leaves a vertex if and only if the vertex has been selected for delivery. Constraint (4.4) guarantees that the number of routes does not exceed the size of the fleet. Constraints (4.5) are the flow conservation constraints. Constraints (4.6) and (4.7) determine the service time at each vertex and ensure that the service time respects the TW of the vertex. Constraints (4.6) also eliminate subtours since they generalize the subtour elimination constraints of Miller-Tucker-Zemlin for the traveling salesman problem (Miller *et al.*, 1960). Constraints (4.8) ensure that all the vehicles return to the depot before the end of its TW. Constraints (4.9) and (4.10) ensure that the capacities of the vehicles are respected. Constraints (4.11) and (4.12) are related to the variable definitions.

The compact formulation above is introduced here to better explain the problem and clarify the constraints. However, this model can be used to solve only very small instances. Therefore, in this paper we present a heuristic column generation approach. It is based on the set covering formulation for the GVRPTW presented in the next section.

4.2.2 A set covering formulation for the GVRPTW

Let Ω denote the set of all feasible routes, i.e., all the routes respecting capacity and time constraints. Let W_r be the cost of route $r \in \Omega$ and let $A_{kr}, \forall k \in \mathcal{K} \setminus \{0\}, r \in \Omega$ indicate whether cluster k is visited on route r ($A_{kr} = 1$) or not ($A_{kr} = 0$). This formulation makes use of binary variables $z_r, \forall r \in \Omega$, that indicate whether route r is selected or not in the solution. The set covering formulation of the GVRPTW is as follows:

$$\text{minimize} \quad \sum_{r \in \Omega} W_r z_r \tag{4.13}$$

$$\text{s.t.} \quad \sum_{r \in \Omega} A_{kr} z_r \geq 1 \qquad \forall k \in \mathcal{K} \setminus \{0\}, \tag{4.14}$$

$$\sum_{r \in \Omega} z_r \leq M, \tag{4.15}$$

$$z_r \in \mathbb{N} \qquad \forall r \in \Omega. \tag{4.16}$$

The objective function (4.13) minimizes the overall delivery costs. Constraints (4.14) ensure that each cluster is visited at least once. Constraints (4.15) ensure that the number of routes in the solution is smaller than the size of the vehicle fleet. Constraints (4.16) are variable definitions. Note that variables are defined as integer to avoid adding constraints $z_r \leq 1, \forall r \in \Omega$ in the linear relaxation. It is clear that all optimal solutions are such that $z_r \leq 1, \forall r \in \Omega$.

Moreover, note that the definition of the problem requires that each cluster is visited exactly once, while Constraints (4.14) require that each cluster is visited at least once. On one hand, if the cost structure satisfies the triangle inequality, it is never optimal to visit a cluster twice. On the other hand, in the LP relaxation of the set covering model, the dual variables associated with Constraints (4.14) are non-negative, which typically leads to a faster convergence of the column generation procedure.

The main drawback of the set covering formulation of the GVRPTW is that it is defined on the complete route set Ω. This set grows exponentially with the number of clusters K. As a consequence, we maintain a subset Ω_1 of Ω that is iteratively populated and on which we solve the set covering formulation.

4.3 Related literature

To the best of our knowledge, there is no existing literature on the GVRPTW. However, there exist works addressing related problems as the generalized VRP (GVRP) and the VRP with roaming delivery locations (VRPRDL). The GVRP is a special case of the GVRPTW where the TWs are not considered. In the VRPRDL, TWs of locations within the same cluster have a specific structure since they do not overlap in time.

The GVRP was introduced by Ghiani & Improta (2000). Customers are associated with multiple service locations and a given demand. The GVRP consists of determining a set of routes for a given number of vehicles with limited capacity such that exactly one service location of each customer is visited. The objective is to minimize the total traveling cost. The GVRP has many applications, such as, urban waste collection problem (Bautista *et al.*, 2008), the vessels routing in maritime transportation, healthcare logistics (Bektaş *et al.*, 2011). Moreover, Baldacci *et al.* (2010) showed that several problems like the traveling salesman problem (TSP) with profits, the VRP with selective backhauls, the covering VRP, and the windy routing problem can be modeled as GVRPs.

Kara & Bektas (2003) proposed a compact integer linear programming formulation for the GVRP, adapting the well-known Miller-Tucker-Zemlin (MTZ) constraints for the TSP to the GVRP. Bektaş *et al.* (2011) proposed four integer linear programming formulations for the GVRP and developed an efficient branch-and-cut algorithm to solve it to optimality. They also developed an adaptive large neighborhood search (ALNS) heuristic to compute upper bounds.

Ha *et al.* (2014) and Afsar *et al.* (2014) studied a variant of the GVRP where the size of the fleet is not fixed. Ha *et al.* (2014) proposed a branch-and-cut algorithm, that provides better results than the one presented by Bektaş *et al.* (2011). Afsar *et al.* (2014) developed a very efficient iterated local search (ILS) which is able to find near optimal solutions in a few seconds.

Zhou *et al.* (2018) introduced a city logistics problem called the multi-depot two-echelon VRP with delivery options. In the second level of the distribution network, two delivery options are considered: customer location and pick-up points. Thus, the second level can be formulated as a GVRP.

Moccia *et al.* (2012) studied what they called the Generalized-VRPTW. This problem differs from the GVRPTW introduced in Section 4.2, since a TW is defined for each cluster while TWs are associated with delivery locations in our case.

The VRPRDL has been introduced by Reyes *et al.* (2017), inspired by the trunk delivery. The objective is to find a minimum-cost set of routes for a fleet of capacitated vehicles in which the order of a customer has to be delivered to the trunk of the customer's car. Deliveries have to take place when the car is parked at one of the locations visited on the customer's travel itinerary. The VRPRDL is a special case of the GVRPTW in which one cluster contains all possible car locations for one customer. The TWs of the locations within a cluster have a specific structure since they are non-overlapping. The authors developed a construction heuristic based on a greedy randomized adaptive search procedure, and an improvement heuristic based on an ALNS. The results highlighted the economic benefits for delivery companies to consider trunk deliveries instead of the traditional home delivery.

Ozbaygin *et al.* (2017) developed a branch-and-price algorithm for the VRPRDL. This algorithm was able to solve to optimality instances with up to 60 clusters in few minutes. For most of the large instances with 120 clusters, the algorithm ended with an optimality gap after 6 hours of computation. Ozbaygin *et al.* (2017) also provided another set of instances for a hybrid delivery strategy combining trunk delivery and

home delivery. In these instances, in each cluster the TW associated with the home location corresponds to the planning horizon and overlaps all other TWs, while the other TW associated with trunk locations are non-overlapping. This instance set also have a specific TW structure. The results revealed that employing this combined strategy led to an average cost savings of nearly 20% with respect to classical delivery system when only home delivery is available.

When the size of the fleet is reduced to a single vehicle, the GVRP and the GVRPTW respectively reduce to the Generalized TSP (GTSP) and the GTSP with TW (GTSPTW). Fischetti *et al.* (1997) proposed an efficient branch-and-cut algorithm to solve the asymmetric version of the GTSP. They developed exact and heuristic separation procedures for some classes of facet-defining inequalities. Yuan *et al.* (2019a) developed a branch-and-cut algorithm for the GTSPTW and proposed several valid inequalities. Their results showed that instances with up to 30 clusters could be solved to optimality within one hour of computation time.

4.4 A column generation based heuristic

In this section we present the algorithm we developed to tackle the GVRPTW. The matheuristic is based on the set covering formulation for the GVRPTW (Section 4.2.2). This formulation relies on an exponential number of variables that represent all the feasible routes. Since generating all routes is not tractable, we use a route pool that is populated along the algorithm. A restricted version of the set covering model is solved considering the routes of the pool only. All the routes inserted into the pool are optimized by applying a procedure described in Section 4.4.1. Given a set of clusters to be visited, the route optimization procedure seeks for the best sequence of clusters, and the best location to visit in each cluster, in order to minimize the cost of the route. The management of the route pool is detailed in Section 4.4.7.

The algorithm is divided in two phases. Phase 1 aims: 1) to construct feasible solutions, and 2) to initialize the pool with promising routes. Phase 1 relies on a construction heuristic (Section 4.4.3) that embeds the route optimization procedure. If the construction heuristic finds a feasible solution, the set covering model is solved in the hope of finding a good combination of the routes generated so far and to improve the current best solution. The solution obtained after solving the set covering model

is improved by applying local search moves (see Section 4.4.5). The above procedure is repeated until the stopping criterion of Phase 1 is reached.

Phase 2 exploits dual information of the linear programming (LP) relaxation of the set covering model to generate additional routes. In particular, the LP relaxation is solved on the current pool of routes. Based on the dual variables, negative reduced cost routes are generated and added to the pool. Then, the set covering model and the local search method are applied, as in Phase 1, in order to improve the current best solution. This procedure is repeated until the stopping criterion of Phase 2 is reached.

In the following, we introduce the main components of the proposed heuristic, i.e., the route optimization procedure in Section 4.4.1, the construction heuristic in Section 4.4.3, the local search procedure in Section 4.4.5 and the negative reduced cost route generation in Section 4.4.6.

4.4.1 The route optimization procedure

In the following, we explain how the route optimization procedure works to optimize a route visiting a set of clusters. This procedure is adapted from the method proposed in Yuan *et al.* (2019a) to obtain an initial solution for the branch-and-cut algorithm that is proposed for the GTSPTW. This procedure can also be viewed as an extension, by taking into account time windows, of the refinement procedure *RP2* proposed in Fischetti *et al.* (1997) for the GTSP.

The route optimization procedure works as follows. Let $\widetilde{\mathcal{K}} \subset \mathcal{K} \setminus \{0\}$ be the set of $n = |\widetilde{\mathcal{K}}|$ different clusters to be visited. We are seeking for a feasible route with a minimum cost that visits all these n clusters. A procedure to generate sequences of clusters is repeated N_{seq} times. Starting from an empty sequence, the sequence of clusters is iteratively generated based on the best insertion concept. Given a sequence $(h_1, \cdots, h_p)$ of p $(p \leq n)$ different clusters of $\widetilde{\mathcal{K}}$ to visit, a labeling algorithm is applied to determine the optimal locations to be visited for each cluster of the sequence $(h_1, \cdots, h_p)$.

Let us first describe the labeling algorithm. Given a sequence $(h_1, \cdots, h_p)$ of p different clusters of $\widetilde{\mathcal{K}}$ to visit, a layered network (LN) is constructed as depicted in Figure 4.2. This network has $p + 2$ layers corresponding to clusters $\mathcal{C}_{h_0} = \mathcal{C}_0$, $\mathcal{C}_{h_1}, \ldots, \mathcal{C}_{h_p}$, $\mathcal{C}_{h_{p+1}} = \mathcal{C}_0$, with their respective vertices. Clusters $\mathcal{C}_{h_0}$ and $\mathcal{C}_{h_{p+1}}$ both represent the depot. The LN contains all arcs (i, j) such that $E_i + T_{ij} \leq L_j$, $i \in \mathcal{C}_{h_f}, j \in \mathcal{C}_{h_{f+1}}, f = 0, \cdots, p$. The objective is to find a path in the LN that starts at $\mathcal{C}_{h_0}$

and arrives at $\mathcal{C}_{h_{p+1}}$ visiting exactly one vertex in each layer, i.e., one vertex from each cluster. If a vertex is visited, the visit must take place during its TW. The solution can be obtained by determining the shortest path with TWs from $\mathcal{C}_{h_0}$ to $\mathcal{C}_{h_{p+1}}$.

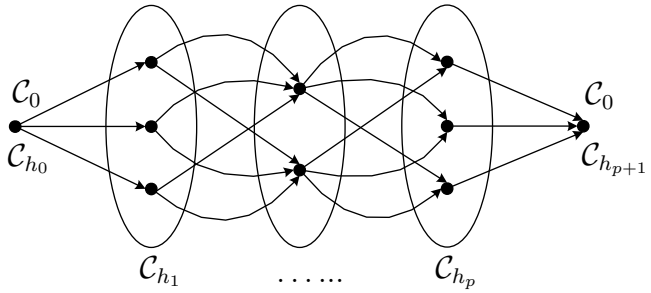


Figure 4.2: The layered network.

To compute the shortest path with TWs on the LN, a labeling algorithm is applied. A label L_i associated with a vertex i consists of a pair (c_i, t_i) representing respectively the cost and service time of a feasible partial path that starts at $\mathcal{C}_{h_0}$ and arrives at vertex i. Let $\mathcal{L}(i)$ be the set containing all the labels associated with vertex i. Suppose that $\mathcal{C}_{\text{cur}}$ is the current cluster and $\mathcal{C}_{\text{pre}}$ is the previous one. First, we calculate the label set $\mathcal{L}(i)$, for all $i \in \mathcal{C}_{\text{cur}}$ by extending labels in $\mathcal{L}(j)$, for all $j \in \mathcal{C}_{\text{pre}}$. Extending a label $L_j \in \mathcal{L}(j)$ towards a vertex $i \in \mathcal{C}_{\text{cur}}$ consists in creating another label $L_i \in \mathcal{L}(i)$ such that:

$$c_i = c_j + C_{ij}, \tag{4.17}$$

$$t_i = \max\{E_i, t_j + T_{ji}\}. \tag{4.18}$$

If $t_i > L_i$, the partial path associated with the label is infeasible and this label is then discarded. In order to make the algorithm efficient, we only keep non-dominated labels. We say that a label L_i^1 dominates a label L_i^2 if and only if $c_i^1 \leq c_i^2$ and $t_i^1 \leq t_i^2$. It is easy to see that extending L_i^1 across the same arcs toward the last vertex of the LN would always produce a better solution than extending L_i^2 in the same way.

Hence, given a fixed visiting sequence of clusters $(h_1, \ldots, h_p)$ of cluster set $\widetilde{\mathcal{K}}$, by applying the labeling algorithm, we can get its corresponding optimal solution. Then, in order to generate good visiting sequences of clusters, we develop a cluster sequence construction procedure based on the best insertion concept. The initial sequence is empty, hence the corresponding LN contains two layers: $\mathcal{C}_{h_0} = \mathcal{C}_{h_1} = \mathcal{C}_0$. Then, at each step, we randomly pick a cluster from $\widetilde{\mathcal{K}}$ that is not yet inserted into the sequence. Suppose the current sequence is $(h_1, \cdots, h_p), p < n$, and cluster $\mathcal{C}_{h_i}$ is chosen to be

inserted next. It is obvious that there are $p+1$ possible insertion positions for index h_i into the sequence. To determine the best insertion position, the labeling algorithm described above is applied $p+1$ times, one for each possible insertion. The sequence that is kept is the one that provides the lowest cost, if such sequence exists. If not, the sequence construction procedure is stopped. The cluster insertion procedure is repeated until $p = n$ to obtain a feasible solution visiting all the clusters in $\widetilde{\mathcal{K}}$. In this process, the labeling algorithm is applied at most $n(n+1)/2$ times.

The sequence construction procedure is repeated N_{seq} times and the best solution is recorded. Note that we always keep the current best solution during the process. During the labeling algorithm, if a label L_i has a cost greater than the cost of the current best solution, then label L_i is discarded. Moreover, note that if we provide an initial feasible sequence of clusters to the route optimization procedure, the current best solution is then initialized with this sequence.

Preliminary experiments lead us to set parameter $N_{seq} = 30$. Note that our algorithm, in the worst case, calls $n(n+1)N_{seq}/2$ times the labeling algorithm. When $n = |\widetilde{\mathcal{K}}| = 5$, the number of all possible cluster sequences is 120, which is less than $6(6+1)30/2 = 630$. It is then more efficient to enumerate all the sequences and to compute the shortest path with TWs on each of them. In this case, the route optimization procedure provides an optimal route that visits the clusters in $\widetilde{\mathcal{K}}$. When $|\widetilde{\mathcal{K}}| = 6$, the number of all possible sequences is 720, which is just a little greater than $6(6+1)30/2 = 630$. Therefore, when $|\widetilde{\mathcal{K}}| < 7$, we choose to proceed with complete enumeration, otherwise we apply the algorithm described above.

4.4.2 Speed-up the route optimization procedure

Since the route optimization procedure need to be called many times through the heuristic, we use dedicated data structure to speed-up computation. In particular, in order to avoid to repeat computations we maintain two pools of cluster sets:

- $\mathcal{CS}_{opt}$ that contains all the cluster sets on which the route optimization procedure has already found a feasible route;

- $\mathcal{CS}_{infea}$ that contains all the cluster sets on which the route optimization procedure has failed to find a feasible solution.

Note that for every route r in the route pool Ω_1, the corresponding cluster set $\mathcal{K}_r$ is in $\mathcal{CS}_{opt}$.

During the construction heuristic (see Section 4.4.3), every time before calling the route optimization procedure for a cluster set $\mathcal{K}_r$, we first detect if the computation has been already performed in the previous iterations. Thus, we check if: 1) $\mathcal{K}_r$ is in $\mathcal{CS}_{opt}$, 2) $\mathcal{K}_r$ is in $\mathcal{CS}_{infea}$, 3) one of the cluster set in $\mathcal{CS}_{infea}$ is the subset of $\mathcal{K}_r$. If condition 1) is true, the best route visiting the clusters in $\mathcal{K}_r$ is already in Ω_1, and we can retrieve the corresponding route. If condition 2) or 3) is true, we know that the route optimization procedure did not find a feasible route that can visit this cluster set. If none of the three conditions is true, then we apply the route optimization procedure to determine the best route visiting the cluster set $\mathcal{K}_r$. Note that in order to be efficient, we use hashing techniques for this implementation.

For the calls to the route optimization procedure outside the construction heuristic, we just detect if the cluster set is already in $\mathcal{CS}_{opt}$. If yes, we just get the corresponding route; otherwise, the route optimization procedure is called.

4.4.3 The construction heuristic

The construction heuristic we developed falls into the category of the parallel insertion heuristics. The number of available vehicles at the depot is denoted as M. The proposed heuristic iteratively construct a set of at most M feasible routes that serve all the customers. Let us denote by $\mathcal{R} = \{r_1, r_2, ..., r_M\}$ the potentially partial routes that we are iteratively constructing. Let $\mathcal{K}_r$ be the index set of customers/clusters served by each route $r \in \mathcal{R}$. Hence, at any step of the construction heuristic, we have $\bigcup_{r\in\mathcal{R}} \mathcal{K}_r \subseteq \mathcal{K}$, and $\mathcal{K}_r \cap \mathcal{K}_{r'} = \emptyset$, for all $r, r' \in \mathcal{R}$. At the end of the algorithm, if $\bigcup_{r\in\mathcal{R}} \mathcal{K}_r = \mathcal{K}$, we found a feasible solution for the GVRPTW.

First, we choose at most M customers relatively difficult to serve. These customers are called *pivot* customers. The different criteria to select pivot customers are described in Section 4.4.3.1. For each pivot customer k, the route optimization procedure (Section 4.4.1) is called to determine the best route r that visits only customer k. This route r is added to the set of partial routes $\mathcal{R}$. If necessary, empty routes are added to $\mathcal{R}$ so that $\mathcal{R}$ contains M partial routes.

Then, an unrouted customers (which is not yet assigned to a route) is selected, as explained hereafter, to be inserted next. For each unrouted customer $k \in \mathcal{K}\backslash(\bigcup_{r\in\mathcal{R}} \mathcal{K}_r)$, we try to insert it into all the partial routes of $\mathcal{R}$. The insertion cost of customer k

into a route r is evaluated using the route optimization procedure. This means that the route may deeply change when inserting a customer. Then we have a precise evaluation of the new cost of the route. The advantage of using the route insertion procedure is detailed in Section 4.4.3.2. The choice of the unrouted customer to insert is performed with a regret strategy. For an unrouted customer k, the regret is defined as the difference between the best and the second best insertion. Note that if for a customer, there exists only one feasible insertion in all the partial routes of $\mathcal{R}$, its regret equals to infinity. Then, the unrouted customer k^* with the maximum regret is chosen to be inserted in the route $r \in \mathcal{R}$ with the minimum insertion cost.

The procedure is repeated until all customers are inserted or insertions are not possible anymore due to time and/or vehicle capacity constraints.

4.4.3.1 Choice of pivot customers

The construction heuristic starts with the choice of pivot customers. These customers may be identified with different criteria. Here we present the three criteria we use in this work. The construction heuristic is called several times during Phase 1. In the first call, Criterion 1 is chosen, while Criterion 2 or Criterion 3 are chosen for the next calls of the construction heuristic. At each call, the choice between Criterion 2 and Criterion 3 is done randomly.

Criterion 1. Due to TWs, some customers cannot be visited on the same route. If customers h and k cannot be served on the same route, we call this pair of customers $\langle h, k\rangle$ *an incompatible pair*. Formally, h and k are incompatible if $E_i + T_{ij} > L_j$ and $E_j + T_{ji} > L_i$, for all $i \in \mathcal{C}_h$, $j \in \mathcal{C}_k$.

Based on all the incompatible pairs, we build a graph $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$, where the vertex set $\bar{\mathcal{V}}$ contains K vertices, one per customer and $(h, k) \in \bar{\mathcal{E}}$ if and only if $\langle h, k\rangle$ is an incompatible pair. We then look for a maximum clique in $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$. By construction of $\bar{\mathcal{G}}$, a clique represents a set of customers such that none of them be visited in the same route since they are all incompatible. Hence, all the customers represented by a clique in $\bar{\mathcal{G}}$ have to be served in different routes. Therefore, we can choose the customers belonging to a maximum clique in $\bar{\mathcal{G}}$ as pivots. Here we use a recursive backtracking algorithm (Carraghan & Pardalos, 1990) that searches for all maximal cliques in graph $\bar{\mathcal{G}}$. It is an enumeration algorithm that backtracks when the size of the current clique plus the size of the set of potential nodes to add is lower than the

size of the current maximum clique. Since the algorithm returns all maximal cliques, one of them is randomly chosen to initialize the pivots customers.

Note that the size of a maximum clique in $\bar{\mathcal{G}}$ can be smaller than M, the number of available vehicles at the depot. In this case, the remaining routes of $\mathcal{R}$ are initialized with empty routes.

Criterion 2. For each vertex $i \in \mathcal{V}$, we determine a vertex set $\mathcal{B}_i = \{j_1, j_2\}$ which includes two vertices compatible with i, and such that j_1 and j_2 do not belong to the same cluster. j_1 is the nearest vertex from which i can be reached (i.e., it satisfies $E_{j_1} + T_{j_1 i} \leq L_i$) and j_2 is the nearest vertex that can be reached from i (i.e., it satisfies $E_i + T_{ij_2} \leq L_{j_2}$). Then, we calculate the average cost $\overline{C_i}$ between vertex i and the vertices in $\mathcal{B}_i$, $\overline{C_i} = \frac{1}{2}\sum_{j \in \mathcal{B}_i} C_{ij}$.

We then define a score w_k for each customer $k \in \mathcal{K} \setminus \{0\}$ as follows:

$$w_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} (C_{0i} + \overline{C_i}). \tag{4.19}$$

By using this score, we select the customers with the highest value of w_k, i.e., customers that are either far away from the depot and/or far away from other customers. However, using only this score has the disadvantage of selecting as pivots some nearby customers which are relatively far from the depot, but that could be served on the same route. Therefore, it is appropriate to *spread* the pivots so that they belong to different spatial regions. To this end, we define $\mathcal{A}_{hk} = \{(i,j) \in \mathcal{A} | i \in \mathcal{C}_h, j \in \mathcal{C}_k\}$ as the set of arcs from $\mathcal{C}_h$ to $\mathcal{C}_k$, and $\overline{C_{hk}}$ as the average traveling cost from cluster $\mathcal{C}_h$ to cluster $\mathcal{C}_k$. $\overline{C_{hk}}$ is calculated as:

$$\overline{C_{hk}} = \frac{1}{|\mathcal{A}_{hk}|} \sum_{(i,j) \in \mathcal{A}_{hk}} C_{ij} \qquad \forall h, k \in \mathcal{K} \setminus \{0\}. \tag{4.20}$$

Let us denote by $\mathcal{P}$ the set of selected pivots. At the beginning, this set $\mathcal{P}$ is empty, and pivots are added one by one. To this end, we define a score w'_k for each customer $k \in \mathcal{K} \setminus \{\mathcal{P} \cup \{0\}\}$ as follows:

$$w'_k = w_k + \min_{h \in \mathcal{P}} \{\min\{\overline{C_{hk}}, \overline{C_{kh}}\}\}. \tag{4.21}$$

The selection of the pivots customers is performed as follows. At first, set $\mathcal{P}$ is empty and we sort all the customers in $k \in \mathcal{K} \setminus \{\mathcal{P} \cup \{0\}\}$ by descending values of w'_k and store them in a list $\mathcal{I}$. Then, we random select a number θ in the interval $[0, 1)$ and calculate θ^ρ. The customer in position $\theta^\rho |\mathcal{I}|$ is chosen as a pivot, and added to $\mathcal{P}$. Then, the scores w'_k of the other customers are updated, and the procedure is repeated until M pivots are selected. Here, we choose $\rho = 6$ as in Ropke & Pisinger

(2006). Note that when $\rho = 1$, the selection becomes purely random. When $\rho = \infty$, the customer associated with the the best score w'_k is selected.

Criterion 3. As with Criterion 2, this criterion computes the scores for each cluster in order to iteratively select the clusters with the highest scores. This criterion is based on a score related to the compactness of the clusters. Suppose that vertex $i \in \mathcal{V}$ has coordinates (a_i, b_i). Here, we consider the barycenter (a^c_k, b^c_k) of a cluster $k \in \mathcal{K} \setminus \{0\}$:

$$a^c_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} a_i, \tag{4.22}$$

$$b^c_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} b_i. \tag{4.23}$$

Then, for each cluster $k \in \mathcal{K} \setminus \{0\}$ we define $\overline{C_{in\mathcal{C}_k}}$ as the average traveling distance of the vertices in $\mathcal{C}_k$ to its barycenter (a^c_k, b^c_k):

$$\overline{C_{in\mathcal{C}_k}} = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \sqrt{(a_i - a^c_k)^2 + (b_i - b^c_k)^2}. \tag{4.24}$$

The score w_k of cluster $k \in \mathcal{K} \setminus \{0\}$ is then defined as:

$$w_k = \sqrt{(a^c_k - a_0)^2 + (b^c_k - b_0)^2} - \overline{C_{in\mathcal{C}_k}}. \tag{4.25}$$

By using this score, we favor compact clusters far from the depot.

Similarly with Criterion 2, we try to spread the pivots. We denote by $\mathcal{P}$ the set of pivots already selected, and we use an updated score w'_k for each cluster $k \in \mathcal{K} \setminus \{\mathcal{P} \cup \{0\}\}$:

$$w'_k = w_k + \min_{h \in \mathcal{P}} \{\min\{\overline{C_{hk}}, \overline{C_{kh}}\}\}. \tag{4.26}$$

Based on w'_k, the selection of pivot customers is performed as described for Criterion 2.

4.4.3.2 Interest in using the route optimization procedure

As mentioned at the beginning of Section 4.4.3, each time we need to compute the insertion cost of a customer in a route, we apply the route optimization procedure described in Section 4.4.1. In the following, we illustrate the advantage of using this optimization procedure instead of a classical best insertion in the context of the GVRPTW.

An illustrative example is provided in Figure 4.3. Let us suppose that during the construction heuristic there is a partial route $r = (0, 2, 4, 7, 0)$ depicted in Figure 4.3(a), visiting a set of clusters $\mathcal{K}_r = \{1, 2, 3\}$. When trying to insert an unrouted cluster $\mathcal{C}_4$, the route optimization method can obtain a best route visiting all the clusters in the

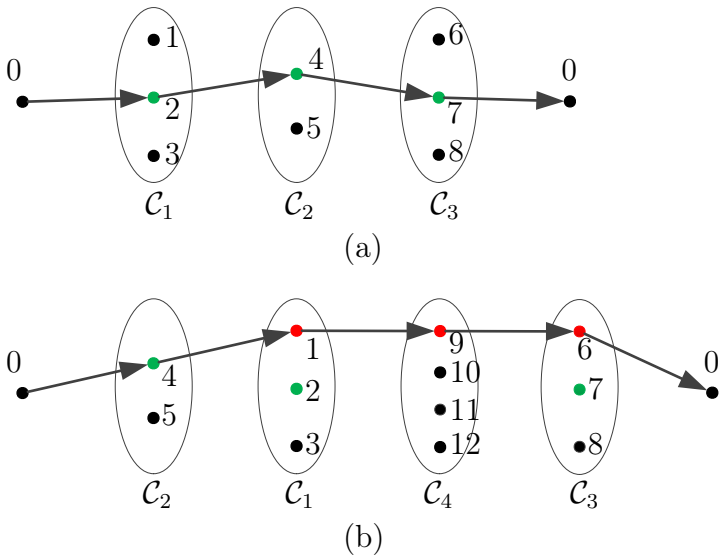


Figure 4.3: Example of route optimization during the construction heuristic.

new set $\mathcal{K}_{r'} = \mathcal{K}_r \cup \{4\} = \{1, 2, 3, 4\}$ as $r' = (0, 4, 1, 9, 6, 0)$, depicted in Figure 4.3(b). By comparing Figure 4.3(a) and Figure 4.3(b), we can see that the visiting sequence of clusters changes from $(1, 2, 3)$ to $(2, 1, 4, 3)$, meanwhile the vertex of each cluster visited in the route r' may be different from the vertices visited in the previous route r, e.g., the vertices visited in clusters $\mathcal{C}_1, \mathcal{C}_3$ change from vertices $2, 7$ (in green) to $1, 6$ (in red) respectively.

In general, in the process of the construction heuristic, when trying to insert an unrouted cluster $\mathcal{C}_k$ into an incumbent route r, the route optimization is used to identify the new best route r' visiting clusters in $\mathcal{K}_r \cup \{k\}$. Then, the cost change when inserting $\mathcal{C}_k$ into r is $\triangle C = W_{r'} - W_r$, where $W_{r'}$ and W_r are the costs of route r' and r respectively.

4.4.4 Recovering infeasibility

When the set covering model is solved on the set of routes Ω_1, the solution that is obtained may visit some clusters more than once. This may occur since if a route r visiting a set of clusters $\mathcal{K}_r$ is added to the route pool Ω_1, there is no guarantee that all the routes visiting subsets of $\mathcal{K}_r$ are also in Ω_1. However, a solution where a cluster is visited more than once is not a feasible solution according to the definition of the GVRPTW given in Section 4.2.

When this occurs, we eliminate the repeated visits of clusters. Let us note Ω_{SC} the set of routes in the solution of the set covering model solved with the route set Ω_1. Let $\mathcal{K}'$ be the set of clusters served in more than one route in set Ω_{SC}. Let us note Ω_{SP} a set of routes generated from the routes in Ω_{SC} as follows. First, Ω_{SP} is initialized with all the routes in Ω_{SC}. For each route r in Ω_{SC}, let us indicate by $\mathcal{K}_r$ the set of clusters visited in r, and by $\widetilde{\mathcal{K}}_r = \mathcal{K}_r \cap \mathcal{K}'$ the set that contains the repeated clusters. Then, for each subset of clusters $\mathcal{S} \subseteq \widetilde{\mathcal{K}}_r$, we consider the cluster set $\mathcal{K}_r \setminus \mathcal{S}$. The route optimization procedure is applied to this cluster set $\mathcal{K}_r \setminus \mathcal{S}$, and the route obtained is stored in Ω_{SP}. Note that, by only keeping vertices in r belonging to clusters in $\mathcal{K}_r \setminus \mathcal{S}$, we may obtain a feasible route $\widetilde{r}$ visiting cluster set $\mathcal{K}_r \setminus \mathcal{S}$. This route $\widetilde{r}$ can be used as an initial solution for the route optimization procedure.

To obtain a feasible solution for the GVRPTW, we solve the set covering model where we replace Constraints (4.14) by $\sum_{r\in\Omega} A_{kr} z_r = 1, \forall k \in \mathcal{K} \setminus \{0\}$. This makes the set covering model a set partitioning model. The new model is then solved on Ω_{SP}. By construction of Ω_{SP}, we have the guarantee to obtain a feasible solution for the GVRPTW.

4.4.5 Local search

A local search method is applied to improve the current best solution obtained after solving the set partitioning model and recovering infeasibility if necessary (Section 4.4.4). The GVRPTW differs from classical vehicle routing problems, since one customer has multiple locations that can be chosen to be visited. In order to improve the flexibility of the local search moves for the GVRPTW, we adopt the so-called enhanced insertion and deletion concept proposed by Reyes *et al.* (2017), and extend it to the swap move.

Figure 4.4 gives an example of the enhanced insertion. A part of an original route is depicted in Figure 4.4(a). The vertices visited on the route are colored in green. When we try to insert a new cluster $\mathcal{C}_u$ between two consecutive clusters $\mathcal{C}_{h_k}$ and $\mathcal{C}_{h_{k+1}}$, a new route can be obtained using the enhanced insertion, as depicted in Figure 4.4(b). The dotted line in Figure 4.4(b) indicates the new route if the classical insertion is used. By comparison, we can see that the enhanced insertion allows the predecessor cluster $\mathcal{C}_{h_k}$ and the successor cluster $\mathcal{C}_{h_{k+1}}$ to change their visited vertices (from green vertices to red vertices).

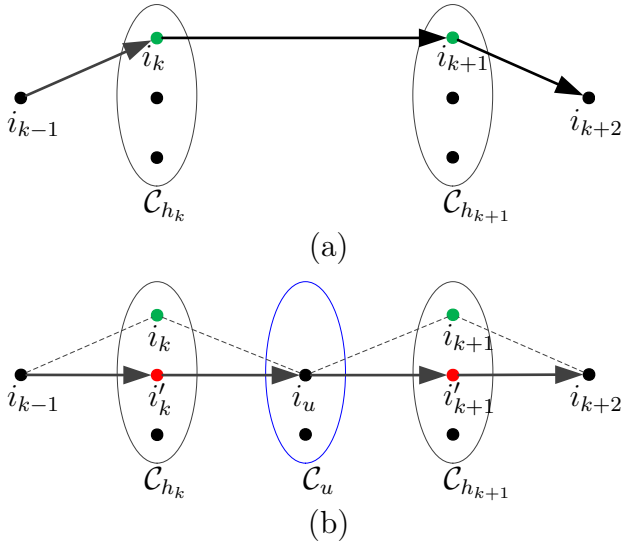


Figure 4.4: An example of enhanced insertion.

The enhanced moves are based on the concept of effective time window. Assume that we have a feasible fixed sequence of clusters $(h_1, \cdots, h_p)$ and a corresponding incumbent feasible route $r = (i_0, i_1, \cdots, i_p, i_{p+1})$, where $i_0 = 0$, $i_k \in \mathcal{C}_{h_k}$, $\forall k \in \{1, \cdots, p\}$, and $i_{p+1} = 0$ is a duplication of the depot. For each i_k in r, we compute its effective TW, $[\xi^r_{i_k}, \zeta^r_{i_k}]$, specifying the earliest and latest times that the vertex i_k can be feasibly visited on this route. Given a route r, these effective TWs are computed using the following recursion rules:

$$\xi^r_{i_0} = E_{i_0}, \xi^r_{i_k} = \max\{E_{i_k}, \xi^r_{i_{k-1}} + T_{i_{k-1}i_k}\} \qquad \forall k \in \{1, \cdots, p+1\}, \tag{4.27}$$

$$\zeta^r_{i_{p+1}} = L_{i_{p+1}}, \zeta^r_{i_k} = \min\{L_{i_k}, \zeta^r_{i_{k+1}} - T_{i_k i_{k+1}}\} \qquad \forall k \in \{0, \cdots, p\}. \tag{4.28}$$

In addition, for each cluster we also keep similar TWs for every vertex $i \in \mathcal{C}_{h_k} \setminus \{i_k\}$:

$$\xi^r_i = \max\{E_i, \xi^r_{i_{k-1}} + T_{i_{k-1}i}\}, \tag{4.29}$$

$$\zeta^r_i = \min\{L_i, \zeta^r_{i_{k+1}} - T_{ii_{k+1}}\}. \tag{4.30}$$

Assuming $\xi^r_i \leq \zeta^r_i$, this TW represents the earliest and latest times to visit $\mathcal{C}_{h_k}$ if the visit happens at the alternate vertex i instead of i_k, with the other vertices of the route unchanged.

With these effective TWs available, we can check whether a new customer u can be inserted on route r at location $i_u \in \mathcal{C}_u$ between customers h_k and h_{k+1} while simultaneously switching customer h_k to delivery location $i \in \mathcal{C}_{h_k}$ and customer h_{k+1} to delivery

location $j \in \mathcal{C}_{h_{k+1}}$ if

$$\max\{E_{i_u}, \xi_i^r + T_{ii_u}\} \leq \min\{L_{i_u}, \zeta_j^r - T_{i_uj}\}. \tag{4.31}$$

Accordingly, the cost increase of the enhanced insertion $\triangle C(h_k, i_u, h_{k+1})$ are computed as follows. Given the incumbent route $r = (0, i_1, \cdots, i_k, i_{k+1}, \cdots, i_p, 0)$, suppose $i'_k \in \mathcal{C}_{h_k} \setminus \{i_k\}$, $i'_{k+1} \in \mathcal{C}_{h_{k+1}} \setminus \{i_{k+1}\}$. The reader can refer to Figure 4.4 for easy understanding of the computation. When a feasible insertion is obtained without changing the predecessor or successor,

$$\triangle C(h_k, i_u, h_{k+1}) = C_{i_k i_u} + C_{i_u i_{k+1}} - C_{i_k i_{k+1}}. \tag{4.32}$$

When a feasible insertion is obtained by only changing the predecessor i_k to i'_k,

$$\triangle C(h_k, i_u, h_{k+1}) = C_{i_{k-1} i'_k} + C_{i'_k i_u} + C_{i_u i_{k+1}} - C_{i_{k-1} i_k} - C_{i_k i_{k+1}}. \tag{4.33}$$

When a feasible insertion is obtained by only changing the successor i_{k+1} to i'_{k+1},

$$\triangle C(h_k, i_u, h_{k+1}) = C_{i_k i_u} + C_{i_u i'_{k+1}} + C_{i'_{k+1} i_{k+2}} - C_{i_k i_{k+1}} - C_{i_{k+1} i_{k+2}}. \tag{4.34}$$

When a feasible insertion is obtained by changing both the predecessor i_k to i'_k and the successor i_{k+1} to i'_{k+1},

$$\begin{aligned}\triangle C(h_k, i_u, h_{k+1}) = {} & C_{i_{k-1} i'_k} + C_{i'_k i_u} + C_{i_u i'_{k+1}} + C_{i'_{k+1} i_{k+2}} \\ & - C_{i_{k-1} i_k} - C_{i_k i_{k+1}} - C_{i_{k+1} i_{k+2}}.\end{aligned} \tag{4.35}$$

The enhanced deletion is illustrated in Figure 4.5. It consists in removing a cluster from the current route allowing to change the vertices visited in the predecessor and successor clusters. The cost variation is computed as previously.

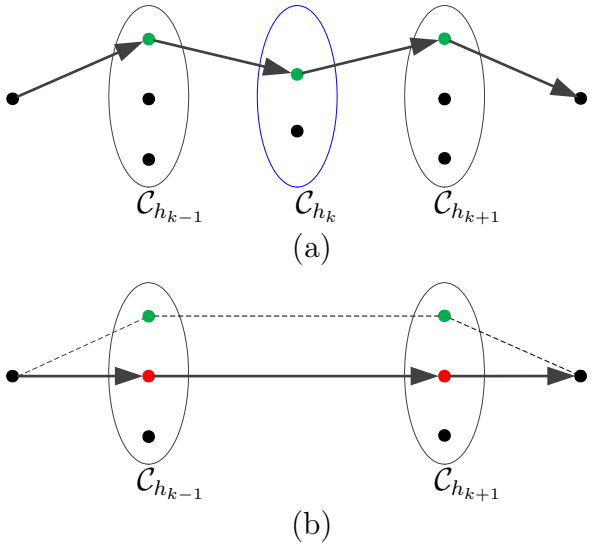


Figure 4.5: An example of enhanced deletion.

Combining the enhanced insertion and deletion, an enhanced relocation is obtained. Moreover, when the relocations of 1) cluster $\mathcal{C}_u$ in the current position of cluster $\mathcal{C}_v$ and 2) cluster $\mathcal{C}_v$ in the current position of cluster $\mathcal{C}_u$ are simultaneously considered, the enhanced swap move is obtained.

In our local search we consider two types of moves: inter-route relocations and swaps. They are sequentially applied to the current best solution. The neighborhoods are completely explored and the best improvement strategy is applied.

Let us indicate by r_1 and r_2 the routes involved in the move, and r'_1 and r'_2 the routes obtained once the move has been applied. Moreover, let us indicate by $\mathcal{K}_{r_1}$, $\mathcal{K}_{r_2}$, $\mathcal{K}_{r'_1}$, $\mathcal{K}_{r'_2}$ the cluster sets associated with routes r_1, r_2, r'_1 and r'_2 respectively.

Before computing the cost increase due to the implementation of a move, we first check if $\mathcal{K}_{r'_1}$ or $\mathcal{K}_{r'_2}$ are present in $\mathcal{CS}_{opt}$. If so, we know the best route that can be obtained from the corresponding cluster set. Then, we just consider the cost of this best route. If the cluster sets $\mathcal{K}_{r'_1}$ or $\mathcal{K}_{r'_2}$ are not present in $\mathcal{CS}_{opt}$, the enhanced local search move is conducted and the cost increase is calculated accordingly. If the total cost increase is negative (the move improves the current solution), then each new route, r'_1 and r'_2, is improved by applying the route optimization procedure if the corresponding cluster set was not present in $\mathcal{CS}_{opt}$.

4.4.6 Negative reduced cost route generation

Let us first introduce some definitions and notations that will be used in this section. The linear relaxation of the set covering model defined by (4.13)∼(4.16) (Section 4.2) is called the master problem (MP). As the size of the set Ω grows exponentially with the number of customers K, the set covering model is usually solved on a subset Ω_1 of Ω and the linear relaxation of the resulting model is called the restricted master problem and denoted as $RMP(\Omega_1)$.

Let λ_k be the nonnegative dual variable associated with the visit of cluster $\mathcal{C}_k$ (Constraints (4.14)), and let λ_0 be the nonpositive dual variable associated with the fleet size constraint (Constraint (4.15)). The dual program $D(\Omega)$ of MP is as follows:

$$\text{maximize} \quad \sum_{k \in \mathcal{K} \setminus \{0\}} \lambda_k + U\lambda_0 \tag{4.36}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K} \setminus \{0\}} A_{kr}\lambda_k + \lambda_0 \leq W_r \qquad \forall r \in \Omega, \tag{4.37}$$

$$\lambda_k \geq 0 \qquad \forall k \in \mathcal{K} \setminus \{0\}, \tag{4.38}$$

$$\lambda_0 \leq 0. \tag{4.39}$$

The reduced cost of a route $r \in \Omega$ as a value

$$W_r - \sum_{k \in \mathcal{K} \setminus \{0\}} A_{kr}\lambda_k - \lambda_0. \tag{4.40}$$

A route associated with a negative reduced cost may reduce the value of the objective function of the $RMP(\Omega_1)$. It is then potentially beneficial to introduce such a route in Ω_1. To generate routes with negative reduced costs, we proceed as follows.

Step 1. For all routes in the route pool Ω_1, we compute their corresponding reduced costs. The routes with reduced costs equal to 0 are collected in a set Ω_1^{rc0}.

Step 2. We apply the deletion, insertion and relocation moves on every route in Ω_1^{rc0}. These moves are based on the enhanced insertion and deletion presented in Section 4.4.5. The routes obtained by applying a move that decreases the reduced cost are stored in a set Γ_{neg}. Let us consider a route $r \in \Omega_1^{rc0}$ and its corresponding cluster set $\mathcal{K}_r$.

- **Deletion**. For each cluster k visited in r, we check if the cluster set $\mathcal{K}_r \setminus \{k\}$ is in $\mathcal{CS}_{opt}$. If not, we apply the enhanced deletion of cluster k from $\mathcal{K}_r$. If we obtain a new route r' which decreases the reduced cost of r, we add r' to set Γ_{neg}.

- **Insertion**. For each cluster k not visited by route r, that is, $k \in \mathcal{K} \setminus \{\mathcal{K}_r \cup \{0\}\}$, we first check if $\mathcal{K}_r \cup \{k\}$ is in $\mathcal{CS}_{opt}$. If not, we apply the enhanced insertion of cluster k into all the possible positions in r. If a new route r' decreases the reduced cost of r, we put r' in Γ_{neg}, and we stop trying to insert cluster k in the remaining positions of route r.

- **Relocation**. For each cluster k that is not visited by route r, i.e., $k \in \mathcal{K} \setminus \{\mathcal{K}_r \cup \{0\}\}$, and for each cluster k' visited by route r, i.e., $k' \in \mathcal{K}_r$ we first check if $\mathcal{K}_r \cup \{k\} \setminus \{k'\}$ is in $\mathcal{CS}_{opt}$. If not, we apply the enhanced relocation of cluster $\mathcal{C}_{k'}$ in the current position of cluster $\mathcal{C}_k$. If a new route r' decreases the reduced cost of r, we add r' to Γ_{neg}.

Step 3. First, we rank all the routes in set Γ_{neg} in ascendant order with respect to their reduced costs. Moreover, all routes in Γ_{neg} are optimized using the route optimization procedure with the original routes as initial solutions. Then, Ω_1^{rc0} is emptied. The first N_{neg}^{best} routes in Γ_{neg} are included in Ω_1^{rc0}, and Γ_{neg} is then emptied.

Step 4. We repeat Step 2 and Step 3 for $Iter_{RC}$ iterations.

4.4.7 Management of the route pool Ω_1

The route pool Ω_1 is a subset of the whole route set Ω. It is populated in the course of the procedure. More specifically, each time the route optimization procedure (explained in Section 4.4.1) finds a feasible route, the route is inserted in Ω_1. This means that, in the construction heuristic, even routes obtained during the evaluation of potential cluster insertions are added to Ω_1 (after a call to the route optimization procedure), even if the insertion is not implemented. Similarly, in the local search method and in the generation of negative reduced cost routes, whenever a move has a negative cost increase, the new routes are added in Ω_1 (after a call to the route optimization procedure), even if the move is not implemented on the current solution.

4.4.8 Overall procedure

Here we explain the overall procedure of the algorithm, and provide some details about the stopping criteria for each phase. In Phase 1, we use the construction heuristic (Section 4.4.3) to build a feasible solution. If a feasible solution is obtained, based on the route pool Ω_1 built so far, we solve the set covering model (Section 4.2.2) to improve the current best solution. A time limit of 30 seconds is set to solve the set covering problem. If the solution visits some clusters more than once, we eliminate the repeated visits to the same cluster (Section 4.4.4). Then, we apply local search moves (Section 4.4.5) to improve the current best solution. If the stopping criteria are not reached, the above procedure is repeated. The stopping criterion of Phase 1 depends on two parameters $Iter1$ and $nbFeaS$. Phase 1 stops after $Iter1$ iterations, or when $nbFeaS$ feasible solutions have been obtained from the construction heuristic.

In Phase 2, the LP relaxation of the set covering model based on the route pool Ω_1 (the restricted master problem $RMP(\Omega_1)$) is solved. Based on the dual information, routes with negative reduced cost are generated (Section 4.4.6). Then, the set covering model and the local search moves are applied as described in Phase 1 trying to improve the current best solution. If the stopping criterion of Phase 2 is not reached, the above procedure is repeated. Phase 2 stops after $Iter2$ iterations, or if the current best solution has not changed in the last $nbNonImprS$ iterations, or if no route with negative reduced cost has been found.

4.5 Computational experiments

In this section, we report the results we obtained with the column generation based heuristic presented in Section 4.4. The algorithm is implemented in C++ and CPLEX 12.6.3 is used to solve linear programs through Concert Technology. All experiments are performed on a machine with Intel(R) Core(TM) i5-6200U CPU, 2.30GHz, 8G RAM.

4.5.1 Instances

In our computational experiments, we consider three sets of instances used by Ozbaygin *et al.* (2017), with 100 instances in total. The first set of instances $\mathcal{S}_1$ consists of 40 VRPRDL instances. For these instances, the number of customers ranges from 15 to 120 and the number of vertices from 50 to 471. Note that due to the presence of TW some vertices cannot be reached. Thus, the number of vertices in the instances can be reduced, ranging from 28 to 293. Each customer has at most 5 delivery locations, with the first and last being home location. The TWs associated with these locations are non-overlapping. The second set of instances $\mathcal{S}_2$ consists of 40 VRPHRDL (VRP with home and roaming delivery locations) instances. Each instance in $\mathcal{S}_2$ is generated from an instance in $\mathcal{S}_1$ by keeping one home location for each customer and replacing its TW with the overall time horizon $[0, T]$. The third set of instances $\mathcal{S}_3$ contains two groups of 10 medium-size VRPRDL instances, all with 40 customers. The number of vertices ranges from 142 to 174, and it can be reduced due to the presence of TW from 90 to 117. The second group consists in locating the delivery locations (except home) closer to the depot of each instance but with shorter TWs.

To test the influence of the route length on the solution, i.e., the number of customers that a vehicle can deliver, on the performance of the proposed algorithm, we modify instances in $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., VRPRDL-variant and VRPHRDL-variant. For an instance of VRPRDL, we first reduce the size of the fleet, the demand of every customer and the traveling time between every two locations by 2 respectively. Whenever the value is not integer, we round it to the smallest integer. Then we modify the TWs of the locations for each customer. The TW of the first location does not change, as well as the upper bounds of the TWs of all the other locations. The lower bound of a location is set to the upper bound of the previous location in the customer's itinerary plus the traveling time to it. The VRPHRDL-variant is generated similarly as the

VRPRDL-variant. For these two sets of instances, the number of vertices ranges from 50 to 471 and it can be reduced due to the presence of TW, from 39 to 392.

4.5.2 Parameters

From the description of the algorithm in Section 4.4, there are six parameters to set. For the stop criterion of Phase 1, there are: $Iter1$ the maximal number of iterations, and $nbFeaS$ the minimal number of feasible solutions to obtain from the construction heuristic. For the stop criterion of Phase 2, there are: $Iter2$ the maximal number of iterations, and $nbNonImprS$ the maximal number of consecutively non-improved solutions. In the negative reduced cost routes generation, there are: N_{neg}^{best} the number of negative reduced cost routes used to iterate the procedure, and $Iter_{RC}$ the number of iterations of the procedure. We conduct the experiments using $Iter1 = 100$, $nbFeaS = 10$, $Iter2 = 10$, $nbNonImprS = 5$, $N_{neg}^{best} = 50$, $Iter_{RC} = 3$.

4.5.3 Preprocessing

The TW width can be reduced by taking into account the earliest and the latest arrival and departure times at each vertex of the graph from or to another vertex. In particular, we consider the following conditions proposed by Desrochers *et al.* (1992):

- earliest arrival time from predecessors: $E_i = \max\{E_i, \min\{L_i, \min_{(j,i)\in\mathcal{A}}(E_j + T_{ji})\}\}$;
- earliest departure time to successors: $E_i = \max\{E_i, \min\{L_i, \min_{(i,j)\in\mathcal{A}}(E_j - T_{ij})\}\}$;
- latest arrival time from predecessors: $L_i = \min\{L_i, \max\{E_i, \max_{(j,i)\in\mathcal{A}}(L_i + T_{ji})\}\}$;
- latest departure time to successors: $L_i = \min\{L_i, \max\{E_i, \max_{(i,j)\in\mathcal{A}}(L_j - T_{ij})\}\}$.

These conditions are applied iteratively to all vertices until no TW can be reduced.

Moreover, we eliminate from graph $\mathcal{G}$ vertices and arcs that cannot be part of any feasible solution. To this end, we:

- eliminate a vertex $i \in \mathcal{V}$ if a round trip from the depot to the vertex, i.e., route $0-i-0$ leads to a time window violation: $E_0 + T_{0i} > L_i$, or $\max\{E_0 + T_{0i}, E_i\} + T_{i0} > L_0$;

- eliminate an arc $(i, j) \in \mathcal{A}$ if it is not possible to go from i to j: $E_i + T_{ij} > L_j$, or if the route that starts from the depot, visits location i, then location j then goes back to the depot (i.e., route $0 - i - j - 0$) violates at least one TW: $\max\{E_0 + T_{0i}, E_i\} + T_{ij} > L_j$, or $\max\{\max\{E_0 + T_{0i}, E_i\} + T_{ij}, E_j\} + T_{j0} > L_0$.

4.5.4 Computational results

The results on instances in $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ are reported in Tables 4.1, 4.2 and 4.3 respectively. We compare the results of the branch-and-price (BP) algorithm presented in Ozbaygin *et al.* (2017) and the column generation based heuristic (CGBH) proposed in this paper. Instances with 15 to 20 customers are called small instances, instances with 30 to 60 customers are called medium-size instances, while instances with 120 customers are large instances.

In Tables 4.1, 4.2 and 4.3, column *instance* represents the name of the instance, column K is the number of customers, and column M is the number of available vehicles at the depot. The next two columns labeled *BP* present the results obtained by the branch-and-price algorithm of Ozbaygin *et al.* (2017). Column *best-known* is the value of the best solution obtained from different parameter settings of the BP. Column *time/s* is the computation time in seconds to obtain the best solution. Note that it does not include the time for the heuristic of Reyes *et al.* (2017) to find an initial solution. The time limit has been set to 2 hours for small and medium-size instances and 6 hours for large instances. If the computation time for one instance is less than the time limit, then it means that the instance was solved to optimality by the BP algorithm. The next columns labeled *CGBH* show the results obtained using the CGBH proposed in this work. Column *obj* represents the objective value obtained, and *time/s* is the computation time in seconds. In the column *GAP/%* we provide the relative gap in percentage between the results obtained from the CGBH and the best-known results provided by the BP algorithm of Ozbaygin *et al.* (2017). It is calculated as $GAP/\% = 100 \times (obj - best\text{-}known)/best\text{-}known$. Column *nbR* represents the number of routes in the best solution obtained by the CGBH. Columns *minL* and *maxL* represent the length of the shortest and longest route in the best solution respectively. Column *averL* is the average length of all the routes in the best solution. Note that in the last row of Tables 4.1 and 4.2, we show the average results restricted to the 10 large instances. In the last row of Table 4.3, we show the average results for all the 20 instances in set $\mathcal{S}_3$.

Table 4.1: Results on VRPRDL instances in set $\mathcal{S}_1$.

instance	K	M	BP		CGBH						
			best-known	time/s	obj	time/s	GAP/%	nbR	minL	maxL	averL
0	15	5	901	0.26	901	0.18	0	4	2	6	3.8
1		6	1286	0.04	1286	0.13	0	5	2	4	3.0
2		5	991	0.07	991	0.16	0	4	3	4	3.8
3		6	1062	0.04	1062	0.11	0	5	1	4	3.0
4		7	1832	0.02	1832	0.14	0	6	1	6	2.5
5	20	6	1294	1.08	1294	0.42	0	5	2	6	4.0
6		5	1155	2.87	1155	1.04	0	4	3	9	5.0
7		7	1455	0.07	1455	0.20	0	6	2	5	3.3
8		6	1260	0.52	1260	0.32	0	5	1	7	4.0
9		8	1684	0.03	1684	0.18	0	7	1	5	2.9
10	30	8	1922	1.13	1922	1.03	0	7	3	7	4.3
11		9	2324	14.61	2324	0.86	0	8	2	6	3.8
12		8	1747	0.68	1747	0.95	0	6	2	10	5.0
13		7	1273	0.64	1273	1.28	0	6	2	6	5.0
14		7	1694	0.50	1694	0.78	0	6	3	7	5.0
15		8	1938	0.75	1938	0.69	0	7	1	7	4.3
16		9	1965	0.73	1965	1.86	0	8	1	11	3.8
17		8	1827	0.23	1827	0.43	0	7	2	6	4.3
18		9	2083	11.13	2083	0.83	0	7	2	7	4.3
19		8	1822	1.53	1822	1.09	0	6	1	8	5.0
20	60	14	3761	4.13	3761	3.87	0	13	2	7	4.6
21		11	2828	10.74	2828	5.82	0	10	3	8	6.0
22		17	4440	1.10	4440	1.78	0	16	2	5	3.8
23		13	3378	11.62	3378	4.81	0	11	2	8	5.5
24		13	3161	643.79	3161	7.46	0	11	2	9	5.5
25		17	4536	1.87	4536	3.03	0	16	1	8	3.8
26		11	2865	7.08	2865	6.05	0	10	2	8	6.0
27		15	4173	43.90	4173	6.19	0	14	1	8	4.3
28		16	3964	38.25	3964	3.42	0	14	2	7	4.3
29		15	4107	1.80	4107	2.11	0	14	2	7	4.3
30	120	19	4935	1629.82	4935	53.79	0	17	2	13	7.1
31		21	5278	21600.00	5267	130.64	**-0.21**	18	3	13	6.7
32		19	5083	21600.00	5061	69.09	**-0.43**	18	1	11	6.7
33		20	5218	8547.16	5218	43.43	0	17	5	11	7.1
34		22	5519	21600.00	5500	62.62	**-0.34**	20	1	14	6.0
35		25	6498	168.13	6498	36.31	0	22	1	9	5.5
36		20	4845	21600.00	4830	38.94	**-0.31**	17	2	11	7.1
37		21	5608	21600.00	5605	51.76	**-0.05**	21	1	12	5.7
38		24	5849	21600.00	5848	61.27	**-0.02**	20	2	9	6.0
39		21	5048	21600.00	5006	73.31	**-0.83**	18	2	11	6.7
Average			**5388.1**	**16154.51**	**5376.8**	**62.11**	**-0.22**	**18.8**	**2.0**	**11.4**	**6.4**

Table 4.2: Results on VRPHRDL instances in set $\mathcal{S}_2$.

instance	K	M	BP		CGBH						
			best-known	time/s	obj	time/s	GAP/%	nbR	minL	maxL	averL
0	15	5	773	0.62	773	0.34	0	3	2	7	5.0
1		6	1065	0.08	1065	0.15	0	4	2	6	3.8
2		5	988	0.11	988	0.33	0	3	3	8	5.0
3		6	914	0.17	914	0.33	0	3	4	6	5.0
4		7	1710	0.04	1710	0.09	0	6	1	5	2.5
5	20	6	1099	2.84	1099	0.95	0	4	2	7	5.0
6		5	996	11.02	996	1.47	0	3	4	12	6.7
7		7	1346	0.33	1346	0.29	0	5	1	6	4.0
8		6	997	0.56	997	0.65	0	4	1	7	5.0
9		8	1166	0.18	1166	0.30	0	4	2	8	5.0
10	30	8	1587	8.54	1587	1.76	0	5	5	9	6.0
11		9	1808	4.7	1808	1.35	0	6	3	9	5.0
12		8	1563	3.38	1563	1.92	0	6	1	10	5.0
13		7	1058	3.26	1058	2.21	0	4	6	9	7.5
14		7	1347	155.93	1347	3.49	0	5	3	8	6.0
15		8	1517	7200.00	1517	2.43	0	5	1	9	6.0
16		9	1445	2.14	1445	2.06	0	5	4	10	6.0
17		8	1627	26.67	1627	2.15	0	5	5	7	6.0
18		9	1461	1.59	1461	1.27	0	5	1	11	6.0
19		8	1715	2.09	1715	1.78	0	6	1	8	5.0
20	60	14	2580	396.47	2580	12.48	0	8	4	9	7.5
21		11	2213	7200.00	2207	35.37	**-0.27**	7	5	14	8.6
22		17	3363	194.98	3363	5.16	0	10	3	8	6.0
23		13	2569	7200.00	2569	17.72	0	8	4	14	7.5
24		13	2400	7200.00	2378	22.45	**-0.92**	8	1	13	7.5
25		17	2845	7200.00	2845	10.91	0	9	3	10	6.7
26		11	2518	33.85	2518	10.33	0	8	4	11	7.5
27		15	2758	3392.94	2758	26.33	0	8	3	15	7.5
28		16	2892	7200.00	2892	15.53	0	9	4	10	6.7
29		15	2691	41.77	2691	6.91	0	8	5	11	7.5
30	120	19	3984	21600.00	3666	123.60	**-7.98**	12	1	16	10.0
31		21	3958	21600.00	3897	308.24	**-1.54**	14	1	15	8.6
32		19	3630	21600.00	3554	319.05	**-2.09**	12	1	15	10.0
33		20	3891	21600.00	3701	194.26	**-4.88**	12	1	18	10.0
34		22	3255	21600.00	3174	277.10	**-2.49**	10	3	16	12.0
35		25	4525	21600.00	4254	287.16	**-5.99**	13	1	14	9.2
36		20	3395	21600.00	3218	153.57	**-5.21**	10	10	16	12.0
37		21	3976	21600.00	3935	241.12	**-1.03**	14	1	13	8.6
38		24	4316	21600.00	4313	132.94	**-0.07**	17	1	15	7.1
39		21	3680	21600.00	3584	408.80	**-2.61**	11	5	17	10.9
Average			**3861.00**	**21600.00**	**3729.60**	**244.58**	**-3.39**	**12.5**	**2.5**	**15.5**	**9.8**

Table 4.1 shows that the proposed CGBH is able to obtain the optimal values for all the small and medium-size instances in less than 10 seconds. For the large instances that were solved to optimality by the BP algorithm (instance 30, 33 and 35), the CGBH gets the optimal values in less than 1 minute. For the large instances that were not solved to optimality, the CGBH always improve the best-known values. For the ten large instances, the CGBH is able to improve by 0.22% the solutions found by the BP algorithm, using around one minute of computation time.

Table 4.2 shows that the CGBH can get the optimal values or even improve the best-known results for all the instances. In some cases (instance 30, 35 and 36), the CGBH improves the best-known values by 7.98%, 5.99% and 5.21% respectively. For the ten large instances, the CGBH averagely improves the best-known results by 3.39% using less than 5 minutes of computation time.

Table 4.3 shows that the CGBH gets the best-known values for all the instances in less than 10 seconds.

Table 4.3: Results on instances in set $\mathcal{S}_3$.

instance	K	M	BP		CGBH						
			best-known	time/s	obj	time/s	GAP/%	nbR	minL	maxL	averL
41_v1	40	11	3203	1249.35	3203	2.34	0	10	2	9	4.0
42_v1		10	2799	3.00	2799	1.35	0	9	1	8	4.4
43_v1		9	2607	7200.00	2607	3.86	0	8	2	12	5.0
44_v1		8	2261	98.52	2261	2.39	0	7	3	7	5.7
45_v1		11	3217	1.63	3217	1.44	0	10	2	7	4.0
46_v1		10	2805	3.81	2805	1.48	0	9	3	7	4.4
47_v1		12	3339	3710.35	3339	2.36	0	10	2	7	4.0
48_v1		11	3325	1.15	3325	1.39	0	10	1	8	4.0
49_v1		12	3534	104.26	3534	1.21	0	11	2	5	3.6
50_v1		10	2752	8.74	2752	4.24	0	10	1	9	4.0
41_v2		8	2133	854.47	2133	7.07	0	7	3	8	5.7
42_v2		8	1946	1005.36	1946	4.11	0	7	1	8	5.7
43_v2		9	1966	270.72	1966	5.29	0	8	2	9	5.0
44_v2		7	1610	41.59	1610	4.91	0	6	1	9	6.7
45_v2		9	2478	9.76	2478	5.83	0	8	2	10	5.0
46_v2		10	2469	27.37	2469	2.39	0	8	3	7	5.0
47_v2		9	1946	68.96	1946	4.55	0	7	2	8	5.7
48_v2		9	2380	477.83	2380	3.01	0	8	3	7	5.0
49_v2		10	2492	13.62	2492	2.41	0	8	2	7	5.0
50_v2		10	2443	164.37	2443	3.69	0	8	3	10	5.0
Average			**2585.25**	**765.74**	**2585.25**	**3.27**	**0**	**8.5**	**2.1**	**8.1**	**4.9**

In conclusion, the proposed CGBH is very efficient. It is able to obtain very high-

quality solutions within short computation times.

The last four columns of each table, i.e., columns *nbR*, *minL*, *maxL*, *averL*, report average statistics on the lengths of the routes in the best solutions. From column *maxL* in Table 4.1 and 4.2, it can be seen that the maximum number of customers that a vehicle serves are 14 and 18 for VRPRDL and VRPHRDL instances respectively. By comparing the average results for the large instances, it can be seen that the average length of routes in the best solution is 6.4 and 9.8 for VRPRDL and VRPHRDL instances respectively. In general, the number of customers that one vehicle serves is not large enough compared with real-life cases. In real life, one courier could make around 30 deliveries per day. In order to be more consistent with real-life cases and enable one vehicle to deliver more customers, we test the CGBH algorithm on the VRPRDL-variant and VRPHRDL-variant instances generated in Section 4.5.1 and the results are reported in Table 4.4.

In Table 4.4, the average results for the medium-size instances are also given. From the average results, for VRPRDL-variant and VRPHRDL-variant instances with 30 customers, the CGBH procedure takes less than half a minute and 1 minute respectively to terminate. For VRPRDL-variant and VRPHRDL-variant instances with 60 customers, it takes around 3 minutes and 6 minutes to terminate respectively. It can be concluded that the CGBH is still very efficient for solving medium-size VRPRDL-variant and VRPHRDL-variant instances. However, for large VRPRDL-variant and VRPHRDL-variant instances with 120 customers, CGBH takes more than 20 minutes and 40 minutes respectively.

However, the length of the routes in the best solution increases compared with the VRPRDL and VRPHRDL instances. For the medium size instances, the maximum number of customers that a vehicle delivers is 24 and 26 for VRPRDL-variant and VRPHRDL-variant respectively. For the large instances, the maximum number of customers that a vehicle serves is 30 and 32 for VRPRDL-variant and VRPHRDL-variant respectively, while the average number becomes 15.9 and 20.0 for VRPRDL-variant and VRPHRDL-variant respectively.

We can conclude that if the number of customers that a vehicle serves becomes large on average, the problem becomes, as expected, more difficult to solve. This is because the route optimization method, which is frequently invoked in the course of the algorithm, becomes more time consuming when the routes are longer.

Table 4.4: Results on variants of VRPRDL and VRPHRDL instances.

instance	K	M	CGBH VRPRDL-variant						CGBH VRPHRDL-variant					
			obj	time/s	nbR	minL	maxL	averL	obj	time/s	nbR	minL	maxL	averL
0	15	3	736	2.90	2	7	8	7.5	675	4.21	2	1	14	7.5
1		3	798	1.98	3	1	10	5.0	745	3.06	2	5	10	7.5
2		3	802	1.94	2	7	8	7.5	796	1.46	2	7	8	7.5
3		3	726	1.20	2	4	11	7.5	708	3.75	2	1	14	7.5
4		4	1125	0.90	2	6	9	7.5	1083	1.70	2	6	9	7.5
5	20	3	951	3.45	3	6	8	6.7	869	6.31	2	6	14	10.0
6		3	831	9.40	2	6	14	10.0	792	11.96	2	7	13	10.0
7		4	869	5.97	2	4	16	10.0	858	9.21	2	6	14	10.0
8		3	893	5.82	2	7	13	10.0	839	8.62	2	8	12	10.0
9		4	929	2.95	2	9	11	10.0	863	5.58	3	1	11	6.7
10	30	4	1129	18.58	2	15	15	15.0	1035	41.96	2	15	15	15.0
11		5	1175	22.17	3	4	13	10.0	1129	30.15	3	4	15	10.0
12		4	1176	17.30	3	7	13	10.0	1121	33.48	3	9	11	10.0
13		4	975	28.45	3	9	12	10.0	953	37.43	3	7	14	10.0
14		4	1079	25.27	2	15	15	15.0	989	45.22	2	14	16	15.0
15		4	1144	23.70	2	13	17	15.0	1023	47.70	2	13	17	15.0
16		5	1224	17.39	3	7	15	10.0	1138	41.21	3	2	18	10.0
17		4	1315	16.26	3	6	12	10.0	1182	44.17	2	12	18	15.0
18		5	1178	21.27	3	8	13	10.0	1021	26.81	2	9	21	15.0
19		4	1181	30.83	2	9	21	15.0	1181	34.68	2	9	21	15.0
Average			**1157.60**	**22.12**	**2.6**	**9.3**	**14.6**	**12.0**	**1077.20**	**38.28**	**2.4**	**9.4**	**16.6**	**13.0**
20	60	7	1874	173.74	4	11	18	15.0	1747	371.10	3	15	23	20.0
21		6	1724	152.21	5	2	20	12.0	1471	343.70	4	1	23	15.0
22		9	1892	106.60	4	14	18	15.0	1717	234.85	4	6	24	15.0
23		7	1976	132.04	5	9	14	12.0	1674	311.15	3	16	22	20.0
24		7	1720	190.23	4	12	18	15.0	1587	457.07	4	2	23	15.0
25		9	1852	192.28	4	13	19	15.0	1632	547.83	3	13	26	20.0
26		6	1749	156.16	4	5	24	15.0	1604	247.60	3	16	24	20.0
27		8	1779	275.09	4	9	22	15.0	1504	437.36	3	18	22	20.0
28		8	1913	121.34	5	1	18	12.0	1742	365.97	4	2	24	15.0
29		8	2017	111.38	4	12	20	15.0	1806	305.60	3	14	25	20.0
Average			**1849.60**	**161.11**	**4.3**	**8.8**	**19.1**	**14.1**	**1648.40**	**362.22**	**3.4**	**10.3**	**23.6**	**18.0**
30	120	10	2471	1243.55	8	1	27	15.0	2244	3016.68	7	1	31	17.1
31		11	2554	1280.57	8	1	23	15.0	2352	2077.37	5	22	29	24.0
32		10	2362	1624.80	7	1	27	17.1	2150	2715.41	6	1	29	20.0
33		10	2702	1340.18	6	14	27	20.0	2468	2798.17	5	16	28	24.0
34		11	2364	1125.76	7	8	24	17.1	2073	2753.88	6	4	29	20.0
35		13	2772	1166.24	7	1	29	17.1	2518	2615.98	6	1	29	20.0
36		10	2535	1123.38	7	6	27	17.1	2250	2739.07	5	14	30	24.0
37		11	2644	1304.54	8	1	24	15.0	2271	2190.87	7	1	31	17.1
38		12	2649	1071.00	12	1	22	10.0	2336	1778.69	9	1	28	13.3
39		11	2529	1305.95	8	1	30	15.0	2255	3085.74	6	1	32	20.0
Average			**2558.20**	**1258.60**	**7.8**	**3.5**	**26.0**	**15.9**	**2291.70**	**2577.19**	**6.2**	**6.2**	**29.6**	**20.0**

4.6 Conclusions

E-commerce is used daily and allows customers to purchase their products online. New last mile delivery services do not require customers to be at a specific location to receive the products they purchase online. Goods can be delivered at home, but as well into lockers, pick-up points or in the trunk of the cars. As a result, and unlike classical vehicle routing problems, several delivery locations are associated with a customer. This new family of delivery problems can be modeled as generalized vehicle routing problems.

In this paper, we have presented the Generalized Vehicle Routing Problem with Time Windows (GVRPTW), and we propose an integer linear programming formulation and a set covering formulation. Based on the set covering formulation, we have developed a column generation based heuristic for the GVRPTW. It combines several components including a construction heuristic, a route optimization procedure, a local search method and a procedure to generate negative reduced cost routes. Computational results on benchmark instances show that the proposed algorithm is very efficient and high-quality solutions can be obtained within very short computation times for instances with up to 120 clusters.

The main perspective of this work is to investigate the dynamic version of routing problem for the last mile delivery services. When some locations or time windows change, a new solution has to be computed again. We believe that the proposed column generation based heuristic could be used in such cases since there are several components to build or optimize solutions, and computation times are short.

Conclusions and perspectives

In this thesis, we studied routing problems arising in the context of the last mile delivery with customers providing multiple shipping locations. The new last mile delivery services do not require customers to go to a specific location to receive the items they purchase online. Packages can be delivered at home, but as well to pick-up points, lockers or in the trunk of the cars. Customers can specify an available time period for each delivery location. As a consequence, and unlike classical vehicle routing problems, several delivery locations are associated with a customer and only one of them needs to be visited during its given time interval to make the delivery. The single-vehicle and multi-vehicle cases of this new family of delivery problems can be modeled as the Generalized Traveling Salesman Problem with Time Windows (GTSPTW) and the Generalized Vehicle Routing Problem with Time Windows (GVRPTW) respectively. The main contributions are as follows.

In Chapter 2, we presented a classification of non-Hamiltonian routing problems including both the single-vehicle case and multi-vehicle case. These problems are characterized by the fact that not all the vertices present in the network need to be visited in the solution.

In Chapter 3, we studied the GTSPTW and proposed four mixed integer linear programming formulations. The dominance relations between the relaxations of these models were established. Two formulations were proved to be superior to the other ones theoretically and experimentally.

In Chapter 4, we proposed a branch-and-cut algorithm to solve the GTSPTW. We presented several problem-specific valid inequalities, which were separated dynamically inside the branch-and-cut algorithm. An initial upper bound was provided by a simple and fast heuristic. Experimental results on different sets of instances showed that the proposed algorithm is effective. Based on the best formulation, instances with up to 30 clusters could be solved to optimality within one hour of computation time.

In Chapter 5, we studied the GVRPTW. A set covering formulation was used to develop a column generation based heuristic to solve this problem. The heuristic combines several components including construction heuristic, route optimization procedure, local search, and the generation of negative reduced cost routes. Experimental results on benchmark instances showed that the proposed algorithm is very efficient and high-quality solutions could be obtained within short computation times for instances with up to 120 clusters.

The topics covered in this thesis provide some management insights for this new last mile delivery service with multiple shipping locations. However, there is still room for new research developments. In the following, we list several research directions that we believe are of interest.

The first perspective is from the methodological point of view. For the GTSPTW, the procedure we have developed to obtain an initial solution is already efficient but could be improved to obtain very high quality solutions. Local search moves or state-of-the art metaheuristics could be adapted for the GTSPTW. For the GVRPTW, in the column generation based heuristic we proposed for the GVRPTW, local search operators are limited. They could be improved by proposing more sophisticated operators dedicated to the GVRPTW. The column generation is applied in a heuristic way, and provides very good upper bounds. It seems interesting to work on the subproblem and the computation of a lower bound.

In this thesis, we assumed that customers provide several possible delivery locations with equal desire to receive the package. Customers do not have preference for a specific location or time period. However in reality, customers might have different preferences for the delivery locations or the delivery time periods they provide. To capture this feature, we could associate each delivery location with a satisfaction factor (profit). Then the new problem could be modeled as a variant of the VRP with profits where customers are associated with several locations or, similarly, as a variant of the GVRP where locations are associated with a profit. The satisfaction/profit factor can appear in the objective function or in the constraints. This problem is of interest because an increasing number of companies are focusing on customer satisfaction to increase the lifetime value of each customer (Kovacs *et al.*, 2014).

In this thesis, customers are associated with several delivery locations that may represent parcel lockers. A locker has a limited capacity that we did not take into account. Due to this fact, it is interesting to work on routing problems with capacity

management associated with delivery locations. In addition, the packages delivered to a locker might not be retrieved by the customer during the same working period. Then the capacity of the locker will be reduced as long as the parcel is not collected by the customer. Therefore, additional constraints have to be considered in the models and algorithms when considering lockers.

Another perspective of this work is to investigate the dynamic version of routing problems in the same context of the last mile delivery with multiple shipping options. When some locations or time windows associated with a customer change during the planning horizon, a new solution has to be recomputed. The column generation based heuristic that we propose for the GVRPTW could be used in such cases since it involves several components to construct or optimize solutions, and the computation times are short.

Appendix A

Proofs for propositions in Chapter 3

Proposition 4.1 *For $i, j \in \mathcal{V} \setminus \{0\}$ suppose that arcs (i, j) and (j, i) are feasible, then*

$$t_i - t_j + T_{ij}x_{ij} + min\{-T_{ji}, E_j - E_i\}x_{ji} \leq L_i y_i - E_j y_j - (L_i - E_j)(x_{ij} + x_{ji}) \quad (3.15)$$

are supervalid inequalities for formulation $\mathcal{F}1$.

Proof. Consider the following constraints

$$t_i - t_j + T_{ij}x_{ij} + \alpha_{ji}x_{ji} \leq L_i y_i - E_j y_j - (L_i - E_j)(x_{ij} + x_{ji}). \quad (A.1)$$

We seek for a value of α_{ji} such that the constraints (A.1) are supervalid, i.e., they are valid for at least one optimal solution.

If $x_{ji} = 0$, these constraints are satisfied for any value of α_{ji}. If $x_{ji} = 1$ (vertex i is visited right after visiting vertex j), then $x_{ij} = 0$, and we obtain constraints $t_i - t_j + \alpha_{ji} \leq 0$, i.e.,

$$\alpha_{ji} \leq t_j - t_i. \quad (A.2)$$

Thus, we seek for a value of α_{ji} such that constraints (A.2) are valid for the values of the decision variables t_j and t_i for at least one optimal solution.

In order to provide such a value, we consider an optimal solution where each vertex is visited as soon as possible. Two cases may be considered.

Case 1: $E_j + T_{ji} \geq E_i$. This means that vertex i can be visited right after vertex j without any waiting time. Hence, we have $t_i = t_j + T_{ji}$. We then obtain:

$$-T_{ji} = t_j - t_i. \quad (A.3)$$

Hence, $\alpha_{ji} = -T_{ji}$ is valid for this first case.

Case 2: $E_j + T_{ji} < E_i$. This means that a waiting time may be required before visiting vertex i. Let us suppose that the value of t_j has been fixed. The value of t_i will then be chosen such that vertex i is visited as soon as possible. If $t_j + T_{ji} \geq E_i$,

then i can be visited right after j without waiting time. Then, similarly to Case 1, $\alpha_{ji} = -T_{ji}$ is valid. If $t_j + T_{ji} < E_i$, then a waiting time is required, and vertex i is visited as soon as possible when $t_i = E_i$. From constraints (3.5), we have $t_j \geq E_j$. We then obtain:

$$E_j - E_i \leq t_j - t_i. \tag{A.4}$$

Hence, when $\alpha_{ji} = E_j - E_i$, constraints (A.2) are valid in this case.

In all cases, we can set $\alpha_{ji} = \min\{-t_{ji}, E_j - E_i\}$. □

Proposition 4.2 *For $h, k \in \mathcal{K} \setminus \{0\}$, suppose that there exists $i \in \mathcal{C}_h$ and $j \in \mathcal{C}_k$ such that the arcs (i,j) and (j,i) are both feasible. Then*

$$\begin{aligned} &\tau_h - \tau_k + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} T_{ij} x_{ij} + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} min\{-T_{ji}, E_j - E_i\} x_{ji} \\ &\leq \sum_{i \in \mathcal{C}_h} L_i y_i - \sum_{j \in \mathcal{C}_k} E_j y_j - \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} (L_i - E_j)(x_{ij} + x_{ji}). \end{aligned} \tag{3.16}$$

are supervalid inequalities for formulation $\mathcal{F}2$.

Proof. Consider the following constraints

$$\begin{aligned} &\tau_h - \tau_k + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} T_{ij} x_{ij} + \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} \alpha_{ji} x_{ji} \\ &\leq \sum_{i \in \mathcal{C}_h} L_i y_i - \sum_{j \in \mathcal{C}_k} E_j y_j - \sum_{\substack{i \in \mathcal{C}_h \\ j \in \mathcal{C}_k}} (L_i - E_j)(x_{ij} + x_{ji}). \end{aligned} \tag{A.5}$$

We seek for a value of α_{ji} such that the constraints (A.5) above are supervalid, i.e., they are valid for at least one optimal solution. If $x_{j'i'} = 0, \forall i' \in \mathcal{C}_h, j' \in \mathcal{C}_k$, these constraints are obviously satisfied for any value of $\alpha_{j'i'}$. Otherwise, we have $x_{ji} = 1$ (vertex i is visited right after visiting vertex j). In this case it follows that:

1. $x_{ij} = 0$, $\forall i \in \mathcal{C}_h, \forall j \in \mathcal{C}_k$;
2. $x_{j'i'} = 0$, $\forall i' \in \mathcal{C}_h \setminus \{i\}, \forall j' \in \mathcal{C}_k \setminus \{j\}$;
3. $y_i = y_j = 1$;
4. $y_l = 0, \forall l \in \mathcal{C}_h \cup \mathcal{C}_k \setminus \{i, j\}$

Hence, we obtain the constraint $\tau_h - \tau_k + \alpha_{ji} \leq 0$, i.e.,

$$\alpha_{ji} \leq \tau_k - \tau_h. \tag{A.6}$$

Thus, we seek for a value of α_{ji} such that constraints (A.6) are valid for the values of the decision variables τ_k and τ_h for at least one optimal solution.

In order to provide such a value, we consider an optimal solution where each vertex is visited as soon as possible. Two cases may be considered.

Case 1: $E_j + T_{ji} \geq E_i$. This means that vertex i can be visited right after vertex j without any waiting time. Hence we have $\tau_h = \tau_k + T_{ji}$. We then obtain

$$-T_{ji} = \tau_k - \tau_h. \tag{A.7}$$

Hence, $\alpha_{ji} = -T_{ji}$ is valid for Case 1.

Case 2: $E_j + T_{ji} < E_i$. This means that a waiting time may be required before visiting vertex i. Let us suppose that the value of τ_k has been fixed. The value of τ_h will then be chosen such that vertex i is visited as soon as possible. If $\tau_k + T_{ji} \geq E_i$, then vertex i can be visited right after j without waiting time. Then, similarly to Case 1, $\alpha_{ji} = -T_{ji}$ is valid. If $\tau_k + T_{ji} < E_i$, then a waiting time is required, and vertex i is visited as soon as possible, i.e., $\tau_h = E_i$. From constraints (3.11), we have $\tau_k \geq E_j$. We then obtain:

$$E_j - E_i \leq \tau_k - \tau_h. \tag{A.8}$$

Hence, $\alpha_{ji} = E_j - E_i$ is valid in this case.

In all cases, we can set $\alpha_{ji} = \min\{-T_{ji}, E_j - E_i\}$. □

Proposition 4.11 *Let $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $\mathcal{C}_k \subseteq \mathcal{S}$, $\pi(\mathcal{C}_k) \neq \emptyset$, the $\pi_{\mathcal{C}_k}$-inequalities:*

$$x((\mathcal{S} \setminus \pi(\mathcal{C}_k)) : (\bar{\mathcal{S}} \setminus \pi(\mathcal{C}_k))) \geq 1 \tag{3.32}$$

are valid for the GTSPTW.
Let $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $\mathcal{C}_k \subseteq \mathcal{S}$, $\sigma(\mathcal{C}_k) \neq \emptyset$, the $\sigma_{\mathcal{C}_k}$-inequalities:

$$x((\bar{\mathcal{S}} \setminus \sigma(\mathcal{C}_k)) : (\mathcal{S} \setminus \sigma(\mathcal{C}_k))) \geq 1 \tag{3.33}$$

are valid for the GTSPTW.
Let $\mathcal{S} \subseteq \mathcal{V} \setminus \{0\}$, $\mathcal{C}_h \prec \mathcal{C}_k$, $\mathcal{C}_h \subseteq \mathcal{S}, \mathcal{C}_k \subseteq \bar{\mathcal{S}}$, $\pi(\mathcal{C}_h) \neq \emptyset, \sigma(\mathcal{C}_k) \neq \emptyset$, the $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequalities:

$$x((\mathcal{S} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k))) : (\bar{\mathcal{S}} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k)))) \geq 1 \tag{3.34}$$

are valid for the GTSPTW.

Proof. (1) Proof for $\pi_{\mathcal{C}_k}$-inequalities. Let tour $\mathcal{T}$ represent a feasible solution. Let $\hat{s}$ be the last vertex of $\mathcal{S}$ visited by $\mathcal{T}$. Since $\mathcal{C}_k \subseteq \mathcal{S}$, $\hat{s} \notin \pi(\mathcal{C}_k)$, so $\hat{s} \in \mathcal{S} \setminus \pi(\mathcal{C}_k)$. Moreover, the successor $\hat{t}$ of $\hat{s}$ in tour $\mathcal{T}$ cannot be in $\pi(\mathcal{C}_k)$, so $\hat{t} \in \bar{\mathcal{S}} \setminus \pi(\mathcal{C}_k)$. Since one vertex in $\mathcal{C}_k$ has to be visited, it is clear that any feasible tour $\mathcal{T}$ contains at least one arc

going from $\mathcal{S} \setminus \pi(\mathcal{C}_k)$ to $\bar{\mathcal{S}} \setminus \pi(\mathcal{C}_k)$. Thus, $\pi_{\mathcal{C}_k}$-inequalities are valid for the GTSPTW. Similarly, $\sigma_{\mathcal{C}_k}$-inequalities can be obtained.

(2) Proof for $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequalities. Let tour $\mathcal{T}$ represent a feasible solution. Let $\hat{s}$ be the last vertex of $\mathcal{S}$ visited by $\mathcal{T}$ and $\hat{t}$ be the first vertex of $\bar{\mathcal{S}}$ visited by $\mathcal{T}$. Since $\mathcal{C}_h \subseteq \mathcal{S}$, $\mathcal{C}_h \prec \mathcal{C}_k$, then $\hat{s} \in \mathcal{S} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k))$. Since $\mathcal{C}_k \subseteq \bar{\mathcal{S}}$, $\mathcal{C}_h \prec \mathcal{C}_k$, then $\hat{t} \in \bar{\mathcal{S}} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k))$. Since $\mathcal{C}_h \prec \mathcal{C}_k$, $\mathcal{C}_h \subseteq \mathcal{S}, \mathcal{C}_k \subseteq \bar{\mathcal{S}}$, it is obvious that any feasible tour $\mathcal{T}$ contains at least one arc going from $\mathcal{S} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k))$ to $\bar{\mathcal{S}} \setminus (\pi(\mathcal{C}_h) \cup \sigma(\mathcal{C}_k))$. Thus, $(\pi_{\mathcal{C}_h}, \sigma_{\mathcal{C}_k})$-inequalities are valid for the GTSPTW. □

Proposition 4.12 *Let us consider $\mathcal{S} \subset \mathcal{V}$ such that $\mathcal{S}$ is infeasible and $|\mathcal{S} \cap \mathcal{C}_k| \leq 1, \forall k \in \mathcal{K}$. For $j \in \mathcal{S}$, let us define $\mathcal{S}(j) = \{i \in \mathcal{C}_{(j)} \setminus \{j\} | \mathcal{S}' = (\mathcal{S} \setminus \{j\}) \cup \{i\}$ is infeasible$\}$. If $|\mathcal{S}(j)| \neq 0$, the clique inequalities (3.35) can be strengthened as:*

$$\sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{S}(j)} y_i \leq |\mathcal{S}| - 1 \qquad \forall j \in \mathcal{S}. \tag{3.36}$$

For $h \in \mathcal{S} \setminus \{j\}$, let us define $\mathcal{S}_j(h) = \{i \in \mathcal{C}_{(h)} \setminus \{h\} | (\mathcal{S} \cup \{j^*, i\}) \setminus \{j, h\}$ is infeasible$, \forall j^* \in \mathcal{S}(j) \cup \{j\}\}$. If $|\mathcal{S}_j(h)| \neq 0$, inequalities above can be lifted as:

$$\sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{S}(j)} y_i + \sum_{i \in \mathcal{S}_j(h)} y_i \leq |\mathcal{S}| - 1 \qquad \forall j, h \in \mathcal{S}. \tag{3.37}$$

Proof. We proof by contradiction. Suppose that $\forall\, j \in \mathcal{S}$,

$$\sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{S}(j)} y_i \geq |\mathcal{S}|. \tag{A.9}$$

$$\implies \sum_{i \in \mathcal{S} \setminus \{j\}} y_i + y_j + \sum_{i \in \mathcal{S}(j)} y_i \geq |\mathcal{S}|. \tag{A.10}$$

$$\implies -\sum_{i \in \mathcal{S} \setminus \{j\}} y_i \leq y_j + \sum_{i \in \mathcal{S}(j)} y_i - |\mathcal{S}|. \tag{A.11}$$

On the other hand by definition of $\mathcal{S}(j)$, we have

$$\sum_{i \in \mathcal{S} \setminus \{j\}} y_i + y_l \leq |\mathcal{S}| - 1 \quad \forall\, l \in \mathcal{S}(j). \tag{A.12}$$

By summing up (A.11) and (A.12), we obtain

$$y_l \leq y_j + \sum_{i \in \mathcal{S}(j)} y_i - 1 \quad \forall\, l \in \mathcal{S}(j). \tag{A.13}$$

Since $\mathcal{S}$ is *infeasible*, $\sum_{i \in \mathcal{S}} y_i \leq |\mathcal{S}| - 1$. According to (A.9), we obtain

$$\sum_{i \in \mathcal{S}(j)} y_i \geq 1. \tag{A.14}$$

Since $\mathcal{S}(j) \subset \mathcal{C}_{(j)}$ and only one vertex in $\mathcal{C}_{(j)}$ is visited, constraint (A.14) means that $\sum_{i \in \mathcal{S}(j)} y_i = 1$, i.e.,

1. $y_j = 0$;
2. $\exists\, l^* \in \mathcal{S}(j)$, $y_{l^*} = 1$;
3. $y_i = 0$, $\forall\, i \in \mathcal{S}(j) \setminus \{l^*\}$.

On the other hand, according to (A.13), we have

$$y_{l^*} \leq 0 \tag{A.15}$$

which is inconsistent with $y_{l^*} = 1$.

Therefore, constraints (3.36) are valid. Similarly, constraints (3.37) can be proved by contradiction. □

A. PROOFS FOR PROPOSITIONS IN CHAPTER 3

Appendix B

Detailed results of Section 3.6.3

We compute the lower bounds at the root node (after applying the preprocessing steps) when just applying one of the families of inequalities. The detailed results for instances in *G1* with at least 7 clusters are provided in Table B.1. We also compute the lower bounds at the root node when only one of the families of inequalities is excluded. The detailed results are reported in Table B.2.

In Table B.1, we give the results under eight different configurations. Columns of *LP* indicate the results obtained by solving the linear relaxation. Columns of *CPLEX* report the results obtained when activating the automatic cut generation of CPLEX (but not the generation of user cuts). The next columns indicate the results obtained by including the automatic cut generation of CPLEX and generating violated inequalities of one of the following families: *Arc Selection*, *Arc-or-Vertex*, *Clique*, *GSEC* and *SOP*. Finally, columns of *Full* indicate the results obtained when applying the automatic cut generation of CPLEX and generating the violated inequalities for all families of valid inequalities.

In Table B.2, we report the results under six different configurations. Columns under *Full* indicate the results obtained by including the automatic cut generation of CPLEX and generating violated inequalities of all families. The next columns show the results obtained by including the automatic cut generation of CPLEX and generating violated inequalities with the exception of one of the following families: *Arc Selection*, *Arc-or-Vertex*, *Clique*, *GSEC* and *SOP*.

Table B.1: Lower bounds obtained at the root node with only one family of valid inequalities.

Instance	LP		CPLEX		Arc selection			Arc-or-vertex			Clique			GSEC			SOP			Full		
	LB	time(s)	Impr(%)	time(s)	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts
TW_7ftv33	324.83	0.03	10.46	0.35	16.63	0.28	15	19.89	0.59	37	10.77	0.33	7	11.63	0.23	4	28.07	0.31	66	28.07	0.25	90
TW_8ftv36	367.33	0.02	9.61	0.28	27.20	0.22	19	13.79	0.32	20	14.55	0.34	15	17.32	0.24	4	39.59	1.29	118	41.70	0.74	107
TW_8ftv38	275.57	0.02	21.21	0.21	39.35	0.17	17	37.50	0.62	164	21.93	0.25	2	39.35	0.50	5	39.35	0.19	33	39.35	0.16	69
TW_9dantzig42	217.23	0.03	4.20	0.25	17.75	0.87	43	4.49	0.33	13	4.52	0.29	2	10.15	0.35	10	35.96	2.33	186	39.73	1.99	257
TW_10att48	2554.36	0.04	20.45	0.58	44.69	0.42	28	29.82	0.53	29	31.93	0.50	4	25.80	0.36	8	61.02	0.90	177	61.02	0.53	151
TW_10gr48	867.60	0.03	19.95	1.00	39.93	0.55	36	18.49	0.72	12	20.36	1.08	0	12.03	0.33	13	65.63	1.80	186	65.63	0.86	258
TW_10hk48	3203.07	0.03	22.14	0.54	35.67	0.73	43	20.30	0.44	28	22.24	0.58	0	22.73	0.35	12	57.50	2.78	230	57.65	1.68	257
TW_11berlin52	2444.10	0.03	6.58	0.59	17.86	0.76	38	19.56	1.16	39	14.73	0.97	8	7.83	0.80	7	33.80	1.79	182	34.21	1.14	186
TW_11eil51	90.28	0.03	21.61	0.71	47.27	1.28	49	14.86	0.61	76	14.64	0.43	4	39.48	0.81	17	67.26	1.82	218	67.26	1.11	273
TW_12brazil58	8021.87	0.08	13.13	0.58	30.28	1.40	58	19.78	0.90	39	14.02	0.72	13	22.85	0.53	11	61.86	2.65	251	64.70	1.31	266
TW_14st70	148.10	0.12	22.87	1.38	47.96	1.22	67	15.39	1.12	18	19.51	1.29	3	24.35	1.33	12	76.68	6.53	469	83.58	5.47	484
TW_16eil76	127.15	0.13	14.31	1.26	28.41	0.95	61	12.61	1.51	91	19.25	1.99	32	15.93	1.23	22	54.41	8.01	477	55.08	4.33	510
TW_16pr76	40353.40	0.12	13.36	1.52	23.51	1.72	65	11.63	1.46	81	17.89	2.38	2	16.26	1.51	16	39.51	11.07	498	40.36	6.50	547
TW_20gr96	18931.60	0.17	17.11	1.82	34.86	2.88	88	22.85	2.43	65	17.40	1.44	90	20.74	1.94	23	67.51	23.77	907	68.71	16.09	1045
TW_20kroA100	6042.69	0.20	16.27	2.67	30.42	3.39	89	13.69	2.84	305	12.82	3.35	65	14.13	1.88	25	54.93	27.43	988	56.09	25.11	1122
TW_20kroB100	6404.64	0.18	15.32	2.30	32.54	3.52	109	9.04	2.22	163	14.37	3.27	8	16.35	2.22	19	46.66	23.40	862	49.61	23.90	1013
TW_20kroC100	5858.21	0.17	7.63	1.79	37.79	2.34	79	7.41	3.10	95	12.38	3.62	60	13.43	2.32	14	61.01	29.26	954	60.77	18.97	1027
TW_20kroD100	5027.16	0.16	9.74	2.10	30.59	2.57	94	8.53	2.35	138	10.35	2.15	5	15.44	2.02	23	74.49	19.86	903	76.22	16.41	901
TW_20kroE100	6119.44	0.22	18.65	2.64	33.75	2.51	97	9.07	2.08	83	14.89	3.28	92	13.25	2.43	30	47.78	21.85	831	50.26	14.81	711
TW_20rat99	258.55	0.16	21.83	2.64	32.59	2.14	80	16.58	2.70	159	19.65	2.46	36	9.59	1.65	38	67.62	20.62	782	72.97	18.33	873
TW_20rd100	2250.39	0.18	14.70	2.55	31.91	2.42	88	12.11	3.14	160	12.47	2.33	2	8.63	2.27	25	48.05	24.57	848	49.40	19.25	797
TW_21eil101	156.36	0.17	4.43	1.57	15.54	2.80	67	9.00	2.74	184	7.67	3.47	18	7.19	2.54	28	48.26	23.08	858	49.08	19.64	857
TW_21lin105	5288.45	0.22	9.47	2.06	23.32	2.93	112	6.31	2.35	142	9.02	2.80	5	14.49	2.58	26	39.07	31.94	1098	40.20	21.40	968
TW_22pr107	10790.70	0.16	9.68	2.43	25.22	2.48	106	9.01	4.06	128	7.42	2.95	9	68.22	2.22	24	96.57	53.01	1663	96.66	17.71	1081
TW_24gr120	1477.08	0.23	16.48	2.94	34.59	3.87	87	9.89	3.33	157	11.10	4.79	9	16.04	2.29	25	80.05	46.33	1216	81.48	37.78	1075
TW_25pr124	19495.90	0.26	10.37	2.94	41.46	2.71	93	9.10	3.92	202	7.16	3.86	76	33.52	2.84	24	74.06	83.20	1856	74.06	71.40	1671
TW_26bier127	53794.50	0.28	5.72	2.75	15.74	4.00	91	17.13	4.38	234	9.15	3.86	25	5.91	2.97	22	30.44	44.56	1031	33.88	15.08	788
TW_26ch130	1882.26	0.28	8.13	3.74	27.98	6.36	137	4.99	5.42	194	9.65	6.52	14	8.90	4.95	29	45.24	78.87	1581	45.39	53.80	1767
TW_28gr137	21929.60	0.31	14.52	4.56	27.44	5.56	109	16.31	6.78	265	11.76	7.05	10	13.34	4.42	27	47.69	73.54	1715	48.94	58.48	1327
TW_28pr136	26733.00	0.27	22.06	4.58	38.50	6.18	131	18.33	7.19	344	20.34	7.27	68	12.91	4.30	30	58.41	73.22	1639	59.31	69.20	1805
TW_29pr144	32337.70	0.30	4.67	3.63	26.35	6.66	154	5.93	5.76	212	5.58	5.20	60	18.92	5.22	25	32.91	192.81	2883	33.43	62.19	1897
TW_30ch150	1891.48	0.35	14.01	5.36	24.06	7.19	121	10.81	6.84	191	12.29	7.29	100	11.12	5.39	32	42.27	99.70	1745	42.71	72.99	1794
TW_30kroA150	6690.17	0.30	12.68	4.79	27.17	7.67	113	12.14	6.41	1008	7.40	4.93	200	12.49	5.62	32	63.10	96.32	1743	63.87	88.09	2199
TW_30kroB150	8089.19	0.35	12.11	5.34	27.49	7.21	139	7.64	6.77	279	9.56	6.20	146	11.42	5.63	41	46.75	125.90	2101	47.87	101.44	1870
TW_31pr152	24678.50	0.33	13.31	3.89	29.28	5.85	121	13.25	4.79	178	16.94	6.73	90	47.12	3.92	37	82.40	77.32	1847	84.58	59.95	2242
TW_32u159	12987.10	0.46	18.49	5.36	40.84	8.12	167	16.48	8.69	475	15.73	7.83	100	18.02	5.54	49	61.85	240.34	2496	63.62	131.26	2233
TW_35si175	4846.48	0.42	3.47	6.48	8.86	9.63	152	3.43	8.88	57	3.52	9.69	0	2.99	6.26	24	11.80	263.54	2466	12.00	180.35	2144
Average		0.18	13.53	2.44	30.13	3.29	85.49	13.71	3.23	163.92	13.65	3.34	37.30	18.38	2.49	21.97	53.77	49.62	1029.84	55.12	33.50	990.86

Table B.2: Lower bounds obtained at the root node without one family of valid inequalities.

Instance	Full			Arc selection			Arc-or-vertex			Clique			GSEC			SOP		
	LB_{full}	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts	Impr(%)	time(s)	nbCuts
TW_7ftv33	416.00	0.25	90	0.00	0.22	59	0.00	0.29	66	0.00	0.26	77	0.00	0.26	87	-3.54	0.42	42
TW_8ftv36	520.52	0.74	107	-0.25	1.04	144	-0.25	1.11	138	-0.25	0.52	96	-0.19	0.71	107	-3.08	0.36	65
TW_8ftv38	384.00	0.16	69	0.00	0.21	70	0.00	0.16	54	0.00	0.16	68	0.00	0.18	56	0.00	0.17	47
TW_9dantzig42	303.53	1.99	257	-2.21	2.28	222	-1.70	1.72	195	-3.13	1.96	244	-0.63	1.59	210	-9.94	0.58	77
TW_10att48	4113.00	0.53	151	0.00	0.67	192	0.00	0.59	148	0.00	0.53	155	0.00	0.56	143	-2.52	0.45	86
TW_10gr48	1437.00	0.86	258	0.00	1.25	372	0.00	0.79	152	0.00	0.84	258	0.00	1.78	299	-7.22	0.64	58
TW_10hk48	5049.67	1.68	257	0.31	2.62	278	-0.57	1.35	178	0.21	1.80	256	0.17	1.90	243	-14.00	0.48	88
TW_11berlin52	3280.19	1.14	186	0.00	1.42	179	-0.02	0.86	139	-0.03	0.64	147	-0.01	0.76	163	-5.49	0.64	95
TW_11eil51	151.00	1.11	273	0.00	1.50	250	0.00	1.04	218	0.00	1.30	236	0.00	1.26	247	-11.14	0.56	100
TW_12brazil58	13212.10	1.31	266	-1.82	1.53	206	-1.66	0.82	221	-1.13	1.28	217	-0.70	1.52	317	-17.29	0.42	102
TW_14st70	271.89	5.47	484	-2.28	6.08	538	0.34	6.13	483	0.21	6.89	522	0.09	5.72	516	-15.98	1.16	99
TW_16eil76	197.18	4.33	510	-1.00	6.22	536	-0.20	5.34	443	0.06	6.09	570	0.04	6.37	473	-13.72	1.55	177
TW_16pr76	56641.00	6.50	547	-0.47	7.39	623	-0.08	7.27	466	0.01	7.03	548	0.10	6.00	512	-10.57	2.30	184
TW_20gr96	31939.90	16.09	1045	-0.65	19.08	963	-0.08	19.22	968	-0.28	16.61	913	0.07	20.71	1021	-16.68	2.39	319
TW_20kroA100	9431.88	25.11	1122	-0.82	24.55	1150	-0.30	20.33	887	-0.18	23.17	1134	-0.06	22.55	1114	-13.23	2.62	327
TW_20kroB100	9581.99	23.90	1013	-0.68	26.28	1012	-0.35	20.17	807	0.15	21.09	874	-0.08	22.16	919	-10.70	2.72	553
TW_20kroC100	9418.48	18.97	1027	-0.09	27.05	1142	0.20	14.97	713	-0.06	17.41	917	-0.08	23.08	1001	-11.82	2.69	288
TW_20kroD100	8858.63	16.41	901	-0.71	20.76	977	0.05	18.83	781	-0.18	19.89	903	-0.26	15.78	961	-15.45	1.92	279
TW_20kroE100	9195.02	14.81	711	-1.58	20.72	845	-0.08	10.87	596	0.01	11.97	686	-0.05	11.55	719	-8.79	2.44	262
TW_20rat99	447.22	18.33	873	-3.09	20.44	928	-0.55	15.83	704	-0.14	16.37	858	-0.31	17.84	967	-19.67	2.13	273
TW_20rd100	3362.12	19.25	797	-0.78	26.64	1033	-0.04	21.76	726	-0.10	18.23	772	0.20	16.26	755	-9.09	3.05	160
TW_21eil101	233.09	19.64	857	-0.56	24.81	941	-0.15	17.62	741	-0.11	18.06	823	0.00	17.70	864	-18.52	2.17	217
TW_21lin105	7414.61	21.40	968	-0.61	28.98	1054	-0.09	17.77	770	0.00	21.64	968	-0.05	22.27	935	-9.33	2.61	195
TW_22pr107	21220.50	17.71	1081	0.19	27.78	1208	-0.13	17.42	881	-0.04	17.84	1031	0.30	33.86	1716	-2.30	4.17	855
TW_24gr120	2680.62	37.78	1075	-0.01	36.94	1253	-0.51	43.01	1014	0.16	26.08	1064	0.24	24.79	1167	-19.66	3.55	266
TW_25pr124	33934.40	71.40	1671	-0.38	83.26	1891	0.09	60.43	1522	0.11	78.95	1628	0.22	64.04	2084	-10.25	5.20	512
TW_26bier127	72021.50	15.08	788	-0.72	29.90	900	-0.10	20.80	723	-0.52	17.33	756	-0.28	20.69	716	-9.15	4.28	388
TW_26ch130	2736.69	53.80	1767	0.02	81.71	1735	0.20	80.18	1499	0.02	61.97	1627	0.31	61.36	1549	-10.04	6.12	319
TW_28gr137	32662.50	58.48	1327	-0.50	56.68	1551	-0.06	50.58	1235	0.20	59.22	1432	0.00	60.78	1363	-12.70	3.89	212
TW_28pr136	42587.60	69.20	1805	-0.84	81.32	1935	-0.49	45.04	1305	-0.01	64.06	1654	-0.13	56.65	1727	-10.09	4.94	454
TW_29pr144	43146.60	62.19	1897	-0.35	134.57	2109	-0.17	58.04	1501	0.03	66.01	1703	-0.23	155.83	2499	-1.50	6.06	478
TW_30ch150	2699.40	72.99	1794	0.27	104.15	2143	-0.06	68.57	1555	0.30	92.57	1585	0.32	80.96	2109	-10.64	6.03	502
TW_30kroA150	10963.00	88.09	2199	-0.16	105.27	2432	-0.16	91.10	1764	-0.06	98.81	2087	0.01	80.42	2329	-17.14	5.89	731
TW_30kroB150	11961.80	101.44	1870	-0.85	105.23	1877	-0.31	93.51	1761	-0.02	88.73	1789	0.21	104.93	1881	-11.87	7.52	571
TW_31pr152	45551.90	59.95	2242	-0.24	81.57	2264	-0.19	64.70	1315	-0.05	53.54	2208	0.02	90.68	2774	-7.94	5.58	655
TW_32u159	21249.90	131.26	2233	-1.20	200.75	2734	-0.22	140.76	2164	0.05	134.66	2163	0.04	164.16	2461	-11.44	7.97	465
TW_35si175	5428.23	180.35	2144	-0.11	289.75	2829	0.00	165.71	1812	0.03	175.55	1893	-0.02	152.90	2054	-2.30	8.30	1293
Average		33.50	990.86	-0.60	45.69	1096.62	-0.21	32.56	828.24	-0.13	33.76	942.22	-0.02	37.04	1055.62	-10.40	3.08	318.76

References

Afsar, H.M., Prins, C. & Santos, A.C. (2014). Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research*, **21**, 153–175. 35, 36, 131

Allahyari, S., Salari, M. & Vigo, D. (2015). A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, **242**, 756–768. 49

Angelelli, E., Archetti, C., Filippi, C. & Vindigni, M. (2017). The probabilistic orienteering problem. *Computers & Operations Research*, **81**, 269–281. 19

Applegate, D., Bixby, R., Chvátal, V. & Cook, W. (2006). *The Traveling Salesman Problem*. Princeton University Press. 11

Aras, N., Aksen, D. & Tekin, M.T. (2011). Selective multi-depot vehicle routing problem with pricing. *Transportation Research Part C: Emerging Technologies*, **19**, 866–884. 43

Archetti, C., Feillet, D., Hertz, A. & Speranza, M.G. (2009). The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, **60**, 831–842. 42

Archetti, C., Bianchessi, N. & Speranza, M.G. (2013). Optimal solutions for routing problems with profits. *Discrete Applied Mathematics*, **161**, 547–557. 42

Archetti, C., Speranza, M.G. & Vigo, D. (2014). Chapter 10: Vehicle routing problems with profits. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, 273–297, SIAM. 16, 17, 18, 42, 46

Archetti, C., Bertazzi, L., Laganà, D. & Vocaturo, F. (2017). The undirected capacitated general routing problem with profits. *European Journal of Operational Research*, **257**, 822–833. 43

Ardalan, Z., Karimi, S., Poursabzi, O. & Naderi, B. (2015). A novel imperialist competitive algorithm for generalized traveling salesman problems. *Applied Soft Computing*, **26**, 546–555. 14

Ascheuer, N., Fischetti, M. & Grötschel, M. (2001). Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, **90**, 475–506. 60, 92, 99

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, **19**, 621–636. 15

Balas, E., Fischetti, M. & Pulleyblank, W.R. (1995). The precedence-constrained asymmetric traveling salesman polytope. *Mathematical programming*, **68**, 241–265. 99, 107

Balcik, B., Beamon, B.M. & Smilowitz, K. (2008). Last mile distribution in humanitarian relief. *Journal of Intelligent Transportation Systems*, **12**, 51–63. 50

Baldacci, R. & Dell'Amico, M. (2010). Heuristic algorithms for the multi-depot ring-star problem. *European Journal of Operational Research*, **203**, 270–281. 53

Baldacci, R., Boschetti, M.A., Maniezzo, V. & Zamboni, M. (2005). Scatter search methods for the covering tour problem. In *Metaheuristic optimization via memory and evolution*, 59–91, Springer. 22

Baldacci, R., Dell'Amico, M. & González, J.S. (2007). The capacitated m-ring-star problem. *Operations research*, **55**, 1147–1162. 50, 52, 54

Baldacci, R., Bartolini, E. & Laporte, G. (2010). Some applications of the generalized vehicle routing problem. *Journal of the operational research society*, **61**, 1072–1077. 37, 130

Baldacci, R., Mingozzi, A. & Roberti, R. (2012). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, **24**, 356–371. 92

Baldacci, R., Hill, A., Hoshino, E.A. & Lim, A. (2017). Pricing strategies for capacitated ring-star problems based on dynamic programming algorithms. *European Journal of Operational Research*, **262**, 879–893. 52

Bautista, J., Fernández, E. & Pereira, J. (2008). Solving an urban waste collection problem using ants heuristics. *Computers & Operations Research*, **35**, 3020–3033. 36, 37, 130

Bektaş, T., Erdoğan, G. & Røpke, S. (2011). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, **45**, 299–316. 35, 36, 37, 126, 130, 131

Ben-Arieh, D., Gutin, G., Penn, M., Yeo, A. & Zverovitch, A. (2003). Transformations of generalized atsp into atsp. *Operations Research Letters*, **31**, 357–365. 13

Ben-Said, A., El-Hajj, R. & Moukrim, A. (2019). A variable space search heuristic for the capacitated team orienteering problem. *Journal of Heuristics*, **25**, 273–303. 45

Bernardino, R. & Paias, A. (2018). Metaheuristics based on decision hierarchies for the traveling purchaser problem. *International Transactions in Operational Research*, **25**, 1269–1295. 30

Bian, Z. & Liu, X. (2018). A real-time adjustment strategy for the operational level stochastic orienteering problem: A simulation-aided optimization approach. *Transportation Research Part E: Logistics and Transportation Review*, **115**, 246–266. 19

Biesinger, B., Hu, B. & Raidl, G.R. (2015). A variable neighborhood search for the generalized vehicle routing problem with stochastic demands. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, 48–60, Springer. 36

Biesinger, B., Hu, B. & Raidl, G. (2016). An integer l-shaped method for the generalized vehicle routing problem with stochastic demands. *Electronic Notes in Discrete Mathematics*, **52**, 245–252. 36

Biesinger, B., Hu, B. & Raidl, G.R. (2018). A genetic algorithm in combination with a solution archive for solving the generalized vehicle routing problem with stochastic demands. *Transportation Science*, **52**, 673–690. 36

Bontoux, B., Artigues, C. & Feillet, D. (2010). A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers & Operations Research*, **37**, 1844–1852. 14

Bovet, J. (1983). The selective travelling salesman problem. 14

BringgTeam (2019). Cutting down on last mile delivery costs: 3 leading strategies. https://www.bringg.com/blog/last-mile/last-mile-delivery-costs/, online, accessed August 2019. 2, 194

Butt, S.E. & Cavalier, T.M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, **21**, 101–111. 42

Calvete, H.I., Galé, C. & Iranzo, J.A. (2013). An efficient evolutionary algorithm for the ring star problem. *European Journal of Operational Research*, **231**, 22–33. 27, 28

Calvete, H.I., Galé, C. & Iranzo, J.A. (2016). Meals: A multiobjective evolutionary algorithm with local search for solving the bi-objective ring star problem. *European Journal of Operational Research*, **250**, 377–388. 27

Carraghan, R. & Pardalos, P.M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, **9**, 375 – 382. 137

Cattaruzza, D., Absi, N., Feillet, D. & González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, **6**. 6

Chao, I.M., Golden, B.L. & Wasil, E.A. (1996a). A fast and effective heuristic for the orienteering problem. *European journal of operational research*, **88**, 475–489. 15

Chao, I.M., Golden, B.L. & Wasil, E.A. (1996b). The team orienteering problem. *European journal of operational research*, **88**, 464–474. 40, 42

Chbichib, A., Mellouli, R. & Chabchoub, H. (2012). Profitable vehicle routing problem with multiple trips: Modeling and variable neighborhood descent algorithm. *American Journal of Operational Research*, **2**, 104–119. 43

Chentli, H., Ouafi, R. & Cherif-Khettaf, W.R. (2018). A selective adaptive large neighborhood search heuristic for the profitable tour problem with simultaneous pickup and delivery services. *RAIRO-Operations Research*, **52**, 1295–1328. 18

Choi, M.J. & Lee, S.H. (2010). The multiple traveling purchaser problem. In *The 40th International Conference on Computers & Indutrial Engineering*, 1–5, IEEE. 54

Cook, W. (2011). *In pursuit of the traveling salesman*. Princeton University Press. 11

Corberán, Á., Plana, I. & Sanchis, J.M. (2012). Arc routing problems: Data instances. https://www.uv.es/corberan/instancias.htm, online, accessed july 2019. 14

Crainic, T., Ricciardi, N. & Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation Science*, **43**, 432–454. 6

Current, J.R. & Schilling, D.A. (1989). The covering salesman problem. *Transportation science*, **23**, 208–213. 21, 24

Current, J.R. & Schilling, D.A. (1994). The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, **73**, 114–126. 23

da Silva Menezes, M., Goldbarg, M.C. & Goldbarg, E.F. (2014). A memetic algorithm for the prize-collecting traveling car renter problem. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, 3258–3265, IEEE. 18

Dantzig, G., Fulkerson, R. & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, **2**, 393–410. 62, 99

Dash, S., Günlük, O., Lodi, A. & Tramontani, A. (2012). A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, **24**, 132–147. 60, 92, 99

Dell'Amico, M., Maffioli, F. & Värbrand, P. (1995). On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, **2**, 297–308. 15

Desrochers, M. & Laporte, G. (1991). Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, **10**, 27–36. 62, 95

Desrochers, M., Desrosiers, J. & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, **40**, 342–354. 78, 103, 148

Dias, T.C.S., de Sousa Filho, G.F., Macambira, E.M., Lucidio dos Anjos, F.C. & Fampa, M.H.C. (2006). An efficient heuristic for the ring star problem. In *International Workshop on Experimental and Efficient Algorithms*, 24–35, Springer. 27

Dimitrijević, V. & Šarić, Z. (1997). An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs. *Information Sciences*, **102**, 105–110. 12, 91

Doerner, K.F. & Hartl, R.F. (2008). Health care logistics, emergency preparedness, and disaster relief: new challenges for routing problems with a focus on the austrian situation. In *The vehicle routing problem: latest advances and new challenges*, 527–550, Springer. 50

Dolinskaya, I., Shi, Z.E. & Smilowitz, K. (2018). Adaptive orienteering problem with stochastic travel times. *Transportation Research Part E: Logistics and Transportation Review*, **109**, 1–19. 19

Drexl, M. (2014). On the generalized directed rural postman problem. *Journal of the Operational Research Society*, **65**, 1143–1154. 14

Dumas, Y., Desrosiers, J., Gelinas, E. & Solomon, M.M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operations research*, **43**, 367–371. 61

El-Hajj, R., Dang, D.C. & Moukrim, A. (2016). Solving the team orienteering problem with cutting planes. *Computers & Operations Research*, **74**, 21–30. 44

eMarketer (2018). Retail ecommerce sales worldwide, 2016-2021 (trillions, % change and % of total retail sales). http://www.emarketer.com/Chart/Retail-Ecommerce-Sales-Worldwide-2016-2021-trillions-change-of-total-retail-sales/215138, online, accessed February 2019. 88

Farber, M. (2016). Consumers are now doing most of their shopping online. http://fortune.com/2016/06/08/online-shopping-increases/, online, accessed March 2019. 1, 88, 193

Feillet, D., Dejax, P. & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation science*, **39**, 188–205. 16, 31

Fischetti, M., Salazar González, J.J. & Toth, P. (1995). The symmetric generalized traveling salesman polytope. *Networks*, **26**, 113–123. 12

Fischetti, M., Salazar González, J.J. & Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, **45**, 378–394. 13, 59, 89, 91, 99, 104, 132, 133

Fischetti, M., Salazar-González, J.J. & Toth, P. (2007). The generalized traveling salesman and orienteering problems. In *The traveling salesman problem and its variations*, 609–662, Springer. 8, 22

Flores-Garza, D.A., Salazar-Aguilar, M.A., Ngueveu, S.U. & Laporte, G. (2017). The multi-vehicle cumulative covering tour problem. *Annals of Operations Research*, **258**, 761–780. 49

Freeman, N.K., Keskin, B.B. & Çapar, İ. (2018). Attractive orienteering problem with proximity and timing interactions. *European Journal of Operational Research*, **266**, 354–370. 19

Gansterer, M., Küçüktepe, M. & Hartl, R.F. (2017). The multi-vehicle profitable pickup and delivery problem. *OR Spectrum*, **39**, 303–319. 44

Gavalas, D., Konstantopoulos, C., Mastakas, K. & Pantziou, G. (2014). A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, **20**, 291–328. 20, 46

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G.E. *et al.* (2019). Efficient cluster-based heuristics for the team orienteering problem with time windows. *APJOR*, **36**, 1950001. 45

Gedik, R., Kirac, E., Milburn, A.B. & Rainwater, C. (2017). A constraint programming approach for the team orienteering problem with time windows. *Computers & Industrial Engineering*, **107**, 178–195. 44

Gendreau, M., Laporte, G. & Semet, F. (1997). The covering tour problem. *Operations Research*, **45**, 568–576. 20, 22

Gendreau, M., Hertz, A., Laporte, G. & Stan, M. (1998). A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, **46**, 330–335. 61

Ghiani, G. & Improta, G. (2000). An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, **122**, 11–17. 33, 35, 92, 130

Goldbarg, M.C., Bagi, L.B. & Goldbarg, E.F.G. (2009). Transgenetic algorithm for the traveling purchaser problem. *European Journal of Operational Research*, **199**, 36–45. 30

Golden, B., Naji-Azimi, Z., Raghavan, S., Salari, M. & Toth, P. (2012). The generalized covering salesman problem. *INFORMS Journal on Computing*, **24**, 534–553. 23

Golden, B.L., Raghavan, S. & Wasil, E.A. (2008). *The vehicle routing problem: latest advances and new challenges*, vol. 43. Springer Science & Business Media. 33

Gomory, R.E. & Hu, T.C. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, **9**, 551–570. 106

Gunawan, A., Lau, H.C. & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, **255**, 315–332. 16, 17, 42

Gutin, G. & Karapetyan, D. (2010). A memetic algorithm for the generalized traveling salesman problem. *Natural Computing*, **9**, 47–60. 14, 91

Gutin, G. & Punnen, A., eds. (2007). *The traveling salesman problem and its variations*. Springer. 11

Ha, M.H., Bostel, N., Langevin, A. & Rousseau, L.M. (2013). An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research*, **226**, 211–220. 48, 49

Ha, M.H., Bostel, N., Langevin, A. & Rousseau, L.M. (2014). An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Computers & Operations Research*, **43**, 9–19. 35, 36, 131

Hachicha, M., Hodgson, M.J., Laporte, G. & Semet, F. (2000). Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, **27**, 29–42. 46, 49

Hamdan, S., Larbi, R., Cheaitou, A. & Alsyouf, I. (2017). Green traveling purchaser problem model: A bi-objective optimization approach. In *2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, 1–6, IEEE. 29

Hamdan, S., Cheaitou, A., Larbi, R. & Alsyouf, I. (2018). Moving toward sustainability in managing assets: A sustainable travelling purchaser problem. In *2018 4th International Conference on Logistics Operations Management (GOL)*, 1–5, IEEE. 29

Hapsari, I., Surjandari, I. & Komarudin, K. (2019). Solving multi-objective team orienteering problem with time windows using adjustment iterated local search. *Journal of Industrial Engineering International*, 1–15. 45

Hawkins, A.J. (2018). Amazon will now deliver packages to the trunk of your car. https://www.theverge.com/2018/4/24/17261744/amazon-package-delivery-car-trunk-gm-volvo, online, accessed May 2019. 1, 40, 59, 125, 194

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, **126**, 106–130. 14, 91

Helsgaun, K. (2009). General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, **1**, 119–163. 14, 91

Helsgaun, K. (2015). Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm. *Mathematical Programming Computation*, **7**, 269–287. 14, 91

Hill, A. & Voss, S. (2016). An equi-model matheuristic for the multi-depot ring star problem. *Networks*, **67**, 222–237. 53

Hodgson, M.J., Laporte, G. & Semet, F. (1998). A covering tour model for planning mobile health care facilities in suhumdistrict, ghama. *Journal of Regional Science*, **38**, 621–638. 24

Holsenbeck, K. (2018). Everything you need to know about amazon locker. https://www.amazon.com/primeinsider/tips/amazon-locker-qa.html, online, accessed May 2019. 1, 125, 193

Honorato, M. (2016). Insights into last mile failed deliveries. https://www.beetrack.com/en/blog/understanding-failed-deliveries, online, accessed August 2019. 2, 194

Hoshino, E.A. & de Souza, C.C. (2008). Column generation algorithms for the capacitated m-ring-star problem. In *International Computing and Combinatorics Conference*, 631–641, Springer. 52

Hoshino, E.A. & De Souza, C.C. (2012). A branch-and-cut-and-price approach for the capacitated m-ring–star problem. *Discrete Applied Mathematics*, **160**, 2728–2741. 52

Hu, W., Fathi, M. & Pardalos, P.M. (2018). A multi-objective evolutionary algorithm based on decomposition and constraint programming for the multi-objective team orienteering problem with time windows. *Applied Soft Computing*, **73**, 383 – 393. 45

Israeli, E. & Wood, R.K. (2002). Shortest-path network interdiction. *Networks: An International Journal*, **40**, 97–111. 65, 96

Jepsen, M.K., Petersen, B., Spoorendonk, S. & Pisinger, D. (2014). A branch-and-cut algorithm for the capacitated profitable tour problem. *Discrete Optimization*, **14**, 78–96. 17

Jozefowiez, N. (2014). A branch-and-price algorithm for the multivehicle covering tour problem. *Networks*, **64**, 160–168. 48

Jozefowiez, N., Semet, F. & Talbi, E.G. (2007). The bi-objective covering tour problem. *Computers & operations research*, **34**, 1929–1942. 23

Kammoun, M., Derbel, H., Ratli, M. & Jarboui, B. (2017). An integration of mixed vnd and vns: the case of the multivehicle covering tour problem. *International Transactions in Operational Research*, **24**, 663–679. 49

Kara, I. & Bektas, T. (2003). Integer linear programming formulation of the generalized vehicle routing problem. In *EURO/INFORMS Joint International Meeting, Istanbul, July*, 06–10. 35, 130

Kara, I., Guden, H. & Koc, O.N. (2012). New formulations for the generalized traveling salesman problem. In *Proceedings of the 6th International Conference on Applied Mathematics, Simulation, Modelling, ASM*, vol. 12, 60–65. 59, 60

Kara, I., Bicakci, P.S. & Derya, T. (2016). New formulations for the orienteering problem. *Procedia Economics and Finance*, **39**, 849–854. 18

Karaoğlan, İ., Erdoğan, G. & Koç, Ç. (2018). The multi-vehicle probabilistic covering tour problem. *European Journal of Operational Research*, **271**, 278–287. 49

Karapetyan, D. (2012). Gtsp instances library. http://www.cs.nott.ac.uk/~pszdk/gtsp.html, online, accessed August 2018. 77, 110

Karapetyan, D. & Gutin, G. (2012). Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. *European Journal of Operational Research*, **219**, 234–251. 91

Kedad-Sidhoum, S. & Nguyen, V.H. (2010). An exact algorithm for solving the ring star problem. *Optimization*, **59**, 125–140. 26, 27

Kirsten, K. (2016). Volvo's solution for the package theft epidemic: Your car's trunk. http://fortune.com/2016/05/10/volvo-urb-it-delivery/, online, accessed March 2019. 1, 39, 59, 125, 193

Kobeaga, G., Merino, M. & Lozano, J.A. (2018). An efficient evolutionary algorithm for the orienteering problem. *Computers & Operations Research*, **90**, 42–59. 18

Kotzeva, M., ed. (2016). *Urban Europe*. Luxembourg: Publications office of the European Union. 6

Kovacs, A.A., Golden, B.L., Hartl, R.F. & Parragh, S.N. (2014). Vehicle routing problems in which consistency considerations are important: A survey. *Networks*, **64**, 192–213. 158

Kovács, G. & Spens, K.M. (2007). Humanitarian logistics in disaster relief operations. *International Journal of Physical Distribution & Logistics Management*, **37**, 99–114. 50

Kubik, P. (2007). *Heuristic solution approaches for the covering tour problem*. Ph.D. thesis, uniwien. 22

Labbé, M. & Laporte, G. (1986). Maximizing user convenience and postal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, **26**, 21–35. 24, 50

Labbé, M., Laporte, G., Martín, I.R. & González, J.J.S. (1999). The median cycle problem. 26

Labbé, M., Laporte, G., Martín, I.R. & Gonzalez, J.J.S. (2004). The ring star problem: Polyhedral analysis and exact algorithm. *Networks: An International Journal*, **43**, 177–189. 26, 27, 28

Labbé, M., Laporte, G., Martin, I.R. & González, J.J.S. (2005). Locating median cycles in networks. *European Journal of Operational Research*, **160**, 457–470. 26

Lahyani, R., Khemakhem, M. & Semet, F. (2013). Heuristics for rich profitable tour problems. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, 1–3, IEEE. 44

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, **43**, 408–416. 33, 55

Laporte, G. & Martín, I.R. (2007). Locating a cycle in a transportation or a telecommunications network. *Networks: An International Journal*, **50**, 92–108. 7, 22, 55, 195

Laporte, G. & Nobert, Y. (1983). Generalized travelling salesman problem through n sets of nodes: An integer programming approach. *INFOR: Information Systems and Operational Research*, **21**, 61–75. 13

Laporte, G. & Semet, F. (1999). Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem. *INFOR: Information Systems and Operational Research*, **37**, 114–120. 12, 91

Laporte, G., Mercure, H. & Nobert, Y. (1987). Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, **18**, 185–197. 13, 14

Laporte, G., Asef-Vaziri, A. & Sriskandarajah, C. (1996). Some applications of the generalized travelling salesman problem. *Journal of the Operational Research Society*, **47**, 1461–1467. 15

Lee, H.K., Ferdinand, F.N., Kim, T.O. & Ko, C.S. (2010). A genetic algorithm based approach to the profitable tour problem with pick-up and delivery. *Industrial Engineering and Management Systems*, **9**, 80–87. 18

Lera-Romero, G. & Miranda-Bront, J.J. (2019). A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints. *European Journal of Operational Research*. 17

Li, K. & Tian, H. (2016). A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. *Applied Soft Computing*, **43**, 469–479. 43, 46

Liefooghe, A., Jourdan, L., Basseur, M., Talbi, E.G. & Burke, E.K. (2008a). Metaheuristics for the bi-objective ring star problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, 206–217, Springer. 27

Liefooghe, A., Jourdan, L., Jozefowiez, N. & Talbi, E.G. (2008b). On the integration of a tsp heuristic into an ea for the bi-objective ring star problem. In *International Workshop on Hybrid Metaheuristics*, 117–130, Springer. 27

Liefooghe, A., Jourdan, L. & Talbi, E.G. (2008c). Metaheuristics and their hybridization to solve the bi-objective ring star problem: a comparative study. *arXiv preprint arXiv:0804.3965*. 27

Liefooghe, A., Jourdan, L. & Talbi, E.G. (2010). Metaheuristics and cooperative approaches for the bi-objective ring star problem. *Computers & Operations Research*, **37**, 1033–1044. 27

Lin, S.W. & Vincent, F.Y. (2017). Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing. *Computers & Industrial Engineering*, **114**, 195–205. 45

Lombard, A., Tamayo-Giraldo, S. & Fontane, F. (2018). Vehicle routing problem with roaming delivery locations and stochastic travel times (vrprdl-s). *Transportation research procedia*, **30**, 167–177. 39

Lopes, R., Souza, V.A. & da Cunha, A.S. (2013). A branch-and-price algorithm for the multi-vehicle covering tour problem. *Electronic Notes in Discrete Mathematics*, **44**, 61–66. 48

Lowe, R. & Rigby, M. (2014). The last mile - exploring the online purchasing and delivery journey. Tech. rep., Barclays. 1, 89, 193

Lu, Y., Benlic, U. & Wu, Q. (2018). A memetic algorithm for the orienteering problem with mandatory visits and exclusionary constraints. *European Journal of Operational Research*, **268**, 54–69. 19

Luis, E., Dolinskaya, I.S. & Smilowitz, K.R. (2012). Disaster relief routing: Integrating research and practice. *Socio-economic planning sciences*, **46**, 88–97. 50

Manerba, D., Mansini, R. & Riera-Ledesma, J. (2017). The traveling purchaser problem and its variants. *European Journal of Operational Research*, **259**, 1–18. 28, 29, 30, 55

Martín-Moreno, R. & Vega-Rodríguez, M.A. (2018). Multi-objective artificial bee colony algorithm applied to the bi-objective orienteering problem. *Knowledge-Based Systems*, **154**, 93–101. 19

Mauttone, A., Nesmachnow, S., Olivera, A. & Robledo, F. (2007). A hybrid metaheuristic algorithm to solve the capacitated m-ring star problem. In *International network optimization conference*, vol. 3. 53

Mei, Y., Salim, F.D. & Li, X. (2016). Efficient meta-heuristics for the multi-objective time-dependent orienteering problem. *European Journal of Operational Research*, **254**, 443–457. 19

Miller, C.E., Tucker, A.W. & Zemlin, R.A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, **7**, 326–329. 8, 10, 38, 61, 94, 129

Moccia, L., Cordeau, J.F. & Laporte, G. (2012). An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society*, **63**, 232–244. 37, 92, 131

Morganti, E., Seidel, S., Blanquart, C., Dablanc, L. & Lenz, B. (2014). The impact of e-commerce on final deliveries: alternative parcel delivery services in france and germany. *Transportation Research Procedia*, **4**, 178–190. 1, 125, 193

Motta, L., Ochi, L.S. & Martinhon, C. (2001). Grasp metaheuristics for the generalized covering tour problem. In *Proceedings of IV metaheuristic international conference, Porto, Portugal*, vol. 1, 387–93. 23

Murakami, K. (2018a). A generalized model and a heuristic algorithm for the large-scale covering tour problem. *RAIRO-Operations Research*, **52**, 577–594. 22

Murakami, K. (2018b). Iterative column generation algorithm for generalized multi-vehicle covering tour problem. *Asia-Pacific Journal of Operational Research*, **35**, 1850021. 50

Naji-Azimi, Z. & Salari, M. (2014). The time constrained maximal covering salesman problem. *Applied Mathematical Modelling*, **38**, 3945–3957. 24

Naji-Azimi, Z., Salari, M. & Toth, P. (2010). A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research*, **207**, 1227–1234. 53

Naji-Azimi, Z., Renaud, J., Ruiz, A. & Salari, M. (2012a). A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *European Journal of Operational Research*, **222**, 596–605. 24, 49, 50

Naji-Azimi, Z., Salari, M. & Toth, P. (2012b). An integer linear programming based heuristic for the capacitated m-ring-star problem. *European Journal of Operational Research*, **217**, 17–25. 53

Nemhauser, G. & Wolsey, L. (1999). Application of special-purpose algorithm. In *Integer and combinatorial optimization*, John Wiley & Sons. 106

Noon, C.E. & Bean, J.C. (1991). A lagrangian based approach for the asymmetric generalized traveling salesman problem. *Operations Research*, **39**, 623–632. 13, 91

Noon, C.E. & Bean, J.C. (1993). An efficient transformation of the generalized traveling salesman problem. *INFOR: Information Systems and Operational Research*, **31**, 39–44. 12, 14, 91

Oliveira, W.A.d., Moretti, A.C. & Reis, E.F. (2015). Multi-vehicle covering tour problem: Building routes for urban patrolling. *Pesquisa Operacional*, **35**, 617–644. 49, 50

Orlis, C., Laganá, D., Dullaert, W. & Vigo, D. (2019). Distribution with quality of service considerations: The capacitated routing problem with profits and service level requirements. *Omega*. 43

Ozbaygin, G. & Savelsbergh, M. (2018). An iterative re-optimization framework for the dynamic vehicle routing problem with roaming delivery locations. 39

Ozbaygin, G., Yaman, H. & Karasan, O.E. (2016). Time constrained maximal covering salesman problem with weighted demands and partial coverage. *Computers & Operations Research*, **76**, 226–237. 22

Ozbaygin, G., Karasan, O.E., Savelsbergh, M. & Yaman, H. (2017). A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, **100**, 115–137. 38, 39, 60, 61, 92, 131, 147, 149

Padberg, M.W. (1973). On the facial structure of set packing polyhedra. *Mathematical programming*, **5**, 199–215. 102

Palomo-Martínez, P.J. & Salazar-Aguilar, M.A. (2019). The bi-objective traveling purchaser problem with deliveries. *European Journal of Operational Research*, **273**, 608–622. 30

Palomo-Martínez, P.J., Salazar-Aguilar, M.A., Laporte, G. & Langevin, A. (2017). A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints. *Computers & Operations Research*, **78**, 408–419. 18, 19

Pandiri, V. & Singh, A. (2019). An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem. *Applied Soft Computing*, **78**, 481–495. 23

Pecin, D., Contardo, C., Desaulniers, G. & Uchoa, E. (2017). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, **29**, 489–502. 66

Pedro, O., Saldanha, R. & Camargo, R. (2013). A tabu search approach for the prize collecting traveling salesman problem. *Electronic Notes in Discrete Mathematics*, **41**, 261–268. 18

Pěnička, R., Faigl, J. & Saska, M. (2019). Variable neighborhood search for the set orienteering problem and its application to other orienteering problem variants. *European Journal of Operational Research*, **276**, 816–825. 20

Pérez, J.A.M., Moreno-Vega, J.M. & Martin, I.R. (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, **151**, 365–378. 27

Pham, T.A., Hà, M.H. & Nguyen, X.H. (2017). Solving the multi-vehicle multi-covering tour problem. *Computers & Operations Research*, **88**, 258–278. 49

Pintea, C.M., Pop, P.C. & Chira, C. (2007). The generalized traveling salesman problem solved with ant algorithms. *Journal of Universal Computer Science*, **13**, 1065–1075. 14

Pop, P.C. (2007). New integer programming formulations of the generalized traveling salesman problem. *American Journal of Applied Sciences*, **4**, 932–937. 59

Pop, P.C., Zelina, I., Lupşe, V., Sitar, C.P. & Chira, C. (2011). Heuristic algorithms for solving the generalized vehicle routing problem. *International Journal of Computers Communications & Control*, **6**, 158–165. 36

Pop, P.C., Kara, I. & Marc, A.H. (2012). New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling*, **36**, 97–107. 35

Pop, P.C., Matei, O. & Sitar, C.P. (2013). An improved hybrid algorithm for solving the generalized vehicle routing problem. *Neurocomputing*, **109**, 76–83. 36

Reihaneh, M. & Ghoniem, A. (2018). A branch-cut-and-price algorithm for the generalized vehicle routing problem. *Journal of the Operational Research Society*, **69**, 307–318. 35

Renaud, J., Boctor, F.F. & Laporte, G. (2004). Efficient heuristics for median cycle problems. *Journal of the Operational Research Society*, **55**, 179–186. 27

Reyes, D., Savelsbergh, M. & Toriello, A. (2017). Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, **80**, 71–91. 38, 39, 60, 61, 92, 111, 126, 131, 141, 149

Riera-Ledesma, J. & Salazar-González, J.J. (2012). Solving school bus routing using the multiple vehicle traveling purchaser problem: A branch-and-cut approach. *Computers & Operations Research*, **39**, 391–404. 30

Ropke, S. & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, **40**, 455–472. 138

Salari, M. & Naji-Azimi, Z. (2012). An integer programming-based local search for the covering salesman problem. *Computers & Operations Research*, **39**, 2594–2602. 23

Salari, M., Reihaneh, M. & Sabbagh, M.S. (2015). Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Computers & Industrial Engineering*, **83**, 244–251. 23

Santini, A. (2019). An adaptive large neighbourhood search algorithm for the orienteering problem. *Expert Systems with Applications*, **123**, 154–167. 18

Saskena, J. (1970a). Mathematical model of scheduling clients through welfare agencies. *Journal of the Canadian Operational Research Society*, **8**, 185–200. 13

Saskena, J.P. (1970b). Mathematical model of scheduling clients through welfare agencies. 14

Shaelaie, M.H., Salari, M. & Naji-Azimi, Z. (2014). The generalized covering traveling salesman problem. *Applied Soft Computing*, **24**, 867–878. 23, 24, 50

Shi, X.H., Liang, Y.C., Lee, H.P., Lu, C. & Wang, Q. (2007). Particle swarm optimization-based algorithms for tsp and generalized tsp. *Information processing letters*, **103**, 169–176. 14

Simonetti, L., Frota, Y. & de Souza, C.C. (2011). The ring-star problem: a new integer programming formulation and a branch-and-cut algorithm. *Discrete Applied Mathematics*, **159**, 1901–1914. 27

Skinderowicz, R. (2018). Solving the uncapacitated traveling purchaser problem with the max–min ant system. In *International Conference on Computational Collective Intelligence*, 257–267, Springer. 30

Smith, S.L. & Imeson, F. (2017). Glns: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, **87**, 1–19. 14, 91

Snyder, L.V. & Daskin, M.S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European journal of operational research*, **174**, 38–53. 14

Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, **35**, 254–265. 110

Srivastava, S., Kumar, S., Garg, R. & Sen, P. (1969). Generalized traveling salesman problem through n sets of nodes. *CORS journal*, **7**, 97. 13, 90, 91

Stavropoulou, F., Repoussis, P.P. & Tarantilis, C.D. (2019). The vehicle routing problem with profits and consistency constraints. *European Journal of Operational Research*, **274**, 340–356. 44

Stenger, A., Schneider, M. & Goeke, D. (2013). The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. *EURO Journal on Transportation and Logistics*, **2**, 57–87. 46

Sun, P., Veelenturf, L.P., Dabia, S. & Van Woensel, T. (2018). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research*, **264**, 1058–1073. 17

Sundar, K. & Rathinam, S. (2017). Multiple depot ring star problem: a polyhedral study and an exact algorithm. *Journal of Global Optimization*, **67**, 527–551. 53

Symonds, D. (2018). Putting a price on failed deliveries. https://www.parcelandpostaltechnologyinternational.com/features/valuing-home-delivery.html, online, accessed August 2019. 2, 194

Tang, L. & Wang, X. (2006). Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. *The International Journal of Advanced Manufacturing Technology*, **29**, 1246–1258. 43, 46

Taniguchi, E., Thompson, R., Yamada, T. & van Duin, R., eds. (2001). *City logistics*. Pergamon, Amsterdam. 6

Toth, P. & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM. 7, 33, 50, 55

Tricoire, F., Graf, A. & Gutjahr, W.J. (2012). The bi-objective stochastic covering tour problem. *Computers & operations research*, **39**, 1582–1592. 48

Tsakirakis, E., Marinaki, M., Marinakis, Y. & Matsatsinis, N. (2019). A similarity hybrid harmony search algorithm for the team orienteering problem. *Applied Soft Computing*, **80**, 776–796. 45

Vansteenwegen, P., Souffriau, W., Berghe, G.V. & Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, **36**, 3281–3290. 45

Vansteenwegen, P., Souffriau, W. & Van Oudheusden, D. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, **209**, 1–10. 16, 17, 42

Varakantham, P., Kumar, A., Lau, H.C. & Yeoh, W. (2018). Risk-sensitive stochastic orienteering problems for trip optimization in urban environments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, **9**, 24. 19

Verbeeck, C., Vansteenwegen, P. & Aghezzaf, E.H. (2016). Solving the stochastic time-dependent orienteering problem with time windows. *European Journal of Operational Research*, **255**, 699–718. 19

Vidal, T., Maculan, N., Ochi, L.S. & Vaz Penna, P.H. (2015). Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science*, **50**, 720–734. 44

Vincent, F.Y., Jewpanya, P., Ting, C.J. & Redi, A.P. (2017). Two-level particle swarm optimization for the multi-modal team orienteering problem with time windows. *Applied Soft Computing*, **61**, 1022–1040. 45

Vincent, F.Y., Jewpanya, P., Lin, S.W. & Redi, A.P. (2019). Team orienteering problem with time windows and time-dependent scores. *Computers & Industrial Engineering*, **127**, 213–224. 45

Wahba, S. (2019). Urban development. https://www.worldbank.org/en/topic/urbandevelopment/overview, online, accessed August 2019. 6

Wardini, L. (2018). 60 stats & trends that will define the future of e-commerce. https://subscriptionly.net/future-of-ecommerce-infographic/, online, accessed Novembre 2018. 1, 124, 193

Xu, J., Chiu, S.Y. & Glover, F. (1999). Optimizing a ring-based private line telecommunication network using tabu search. *Management Science*, **45**, 330–345. 54

Yang, J., Shi, X., Marchese, M. & Liang, Y. (2008). An ant colony optimization method for generalized tsp problem. *Progress in Natural Science*, **18**, 1417–1422. 14

Yu, Q., Fang, K., Zhu, N. & Ma, S. (2019). A matheuristic approach to the orienteering problem with service time dependent profits. *European Journal of Operational Research*, **273**, 488–503. 20

Yuan, Y., Cattaruzza, D., Ogier, M. & Semet, F. (2019a). A branch-and-cut algorithm for the generalized traveling salesman problem with time windows, submitted paper. 66, 77, 132, 133

Yuan, Y., Cattaruzza, D., Ogier, M. & Semet, F. (2019b). A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for routing problems with time windows, submitted paper. 95, 96

Yuan, Y., Cattaruzza, D., Ogier, M., Semet, F. & Vigo, D. (2019c). The generalized vehicle routing problem with time windows. In *VeRoLog 2019 Workshop of the EURO Working Group on Vehicle Routing and Logistics optimization, June 2019*. 66

Zhang, T., Chaovalitwongse, W.A., Zhang, Y.J. & Pardalos, P.M. (2009). The hot-rolling batch scheduling method based on the prize collecting vehicle routing problem. *Journal of Industrial and Management Optimization*, **5**, 749–765. 43

Zhang, Z., Qin, H. & Lim, A. (2014). A memetic algorithm for the capacitated m-ring-star problem. *Applied intelligence*, **40**, 305–321. 53

Zhou, L., Baldacci, R., Vigo, D. & Wang, X. (2018). A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. *European Journal of Operational Research*, **265**, 765–778. 36, 131

Zia, M., Cakir, Z. & Seker, D. (2018). Spatial transformation of equality–generalized travelling salesman problem to travelling salesman problem. *ISPRS International Journal of Geo-Information*, **7**, 115. 13

Résumé étendu en Français

De nos jours, le commerce électronique est un marché florissant dans le monde entier. Il est utilisé quotidiennement et permet aux clients de faire leurs achats en ligne, quand ils le veulent. Les clients ne sont plus obligés de se rendre dans un magasin spécifique et de respecter les heures d'ouverture. Un sondage annuel mené par la société d'analyse comScore et UPS a révélé que les consommateurs américains achetaient plus d'articles en ligne qu'en magasin en 2016 (Farber, 2016). À la fin de l'année 2018, les ventes mondiales du commerce électronique atteignaient environ 2,8 milliards de dollars et devraient atteindre 4,5 milliards de dollars en 2021 (Wardini, 2018). La croissance très importante des ventes par le commerce électronique pose un énorme défi pour la livraison du dernier kilomètre, car les articles commandés doivent être livrés individuellement à chaque client.

Il existe actuellement plusieurs services de livraison du dernier kilomètre permettant de livrer des colis aux clients. L'option de livraison la plus courante est la livraison à domicile ou au travail (Lowe & Rigby, 2014). Les clients attendent chez eux ou sur leur lieu de travail pour recevoir leurs colis. En outre, la livraison peut être effectuée à des points de collecte tels que des consignes (*lockers*) ou des magasins. Dans ce cas, les clients peuvent récupérer leurs colis après la livraison. Pour donner une idée, il existe plus de 2800 consignes Amazon situées aux États-Unis (Holsenbeck, 2018). Lorsque les clients achètent en ligne, ils peuvent choisir une consigne à proximité comme lieu de livraison. Cela réduit la fragmentation des livraisons sur le dernier kilomètre, contribuant ainsi à réduire les encombrements et la pollution de l'environnement causés par le transport de marchandise en milieu urbain (Morganti *et al.*, 2014), ainsi que les coûts de livraison. Ces dernières années, un nouveau concept appelé *livraison dans le coffre / dans la voiture* a été proposé. Ici, les colis des clients peuvent être livrés directement dans les coffres des voitures. Volvo a lancé le premier service au monde de livraison dans les voitures en Suède en 2016 (Kirsten, 2016). Le livreur a un code

numérique unique pour accéder au coffre de la voiture. En avril 2018, Amazon a lancé le service en partenariat avec deux grands constructeurs, General Motors et Volvo. Ce service est disponible dans 37 villes aux États-Unis (Hawkins, 2018). La livraison dans le coffre diffère de la livraison à domicile ou au travail et des points de collecte, car la voiture se déplace et peut se trouver à différents endroits pendant différentes périodes. Par exemple, elle reste garée sur le lieu de travail le matin et sur le parking d'un centre commercial en fin d'après-midi. En conséquence, la synchronisation entre la voiture et le livreur est nécessaire pour effectuer la livraison.

Tous ces services de livraison peuvent être combinés et, au lieu de choisir un seul lieu de livraison lors d'un achat en ligne, le client peut proposer un ensemble de lieux de livraison avec les contraintes de temps associées. Pour livrer un colis à un client donné, le livreur dit alors choisir l'un des emplacements fourni par le client.

Dans cette thèse, notre objectif est de modéliser et de développer des méthodes de résolution efficaces des problèmes de tournées de véhicules dans le contexte de la livraison du dernier kilomètre offrant plusieurs options de livraison: à domicile , sur le lieu de travail, en points de collecte et dans le coffre de la voiture. La livraison du dernier kilomètre avec plusieurs options d'expédition permet aux clients de choisir plusieurs emplacements pour recevoir leurs colis. Cela offre aux clients plus de flexibilité en tenant compte de leur convenance. En outre, cela peut augmenter le taux de première livraison réussie et ainsi permettre de réduire les coûts de livraison. Au Royaume-Uni, par exemple, le coût des livraisons ayant échoué s'élève à près de 1,1 milliard de dollars pour les détaillants et les entreprises de commerce électronique sur un marché de 100 milliards de dollars (Honorato, 2016; Symonds, 2018). Offrir plus d'options de livraison peut être rentable (BringgTeam, 2019).

Nous étudions les problèmes de livraison avec une flotte composée d'un seul véhicule, et avec une flotte composée de plusieurs véhicules, c'est-à-dire le problème du voyageur de commerce généralisé avec fenêtres de temps (GTSPTW) et le problème de tournées de véhicules généralisé avec fenêtres de temps (GVRPTW). Dans ces problèmes, les différents emplacements de livraison possibles associés à un même client sont regroupés au sein d'un cluster. Il est facile de constater que, dans les problèmes étudiés, il n'est pas nécessaire de visiter tous les lieux associés à un client, car le livreur n'a besoin de livrer le colis qu'à un seul des emplacements fournis par le client.

Dans ce qui suit, nous résumons les travaux réalisés dans cette thèse.

Dans le chapitre 1, nous présentons une classification des problèmes de routage non hamiltoniens (non-HRP), caractérisés par le fait que tous les sommets présents dans le graphe ne doivent pas nécessairement être visités pour qu'une solution soit réalisable. Laporte & Martín (2007) proposent une revue de la littérature sur cette thématique, limitée aux problèmes où la flotte est composée d'un seul véhicule, et couvrant les travaux publiés jusqu'en 2005. Nous proposons une revue de la littérature sur les progrès récents dans les non-HRPs avec un seul véhicule et avec plusieurs véhicules. Les problèmes concernant un seul véhicule incluent le *Generalized Traveling Salesman Problem (GTSP)*, le *Traveling Salesman Problem with Profits (TSPPs)*, le *Covering Tour Problem (CTP)*, le *Covering Salesman Problem (CSP)*, le *Median Cycle Problem (MCP)*, le *Ring Star Problem (RSP)* et le *Traveling Purchaser Problem (TPP)*. Les problèmes avec plusieurs véhicules incluent le *Generalized Vehicle Routing Problem (GVRP)*, le *Generalized Vehicle Routing Problem with Time Windows (GVRPTW)*, le *Vehicle Routing Problem with Profits (VRPPs)*, le *multi-vehicle CTP (*mCTP*), le capacitated multiple RSP (*CmRSP*) et le* multi-vehicle TPP (*mTPP*). Pour chacun de ces problèmes, nous présentons sa définition, une formulation mathématique compacte, une revue de la littérature et certaines de leurs applications.

Dans le chapitre 2, nous étudions le problème de la livraison du dernier kilomètre avec de multiples options de livraison dans le cas avec un seul véhicule. Ce problème est modélisé comme le problème du voyageur de commerce généralisé avec fenêtres de temps (GTSPTW). Le GTSPTW est défini sur un graphe orienté dans lequel l'ensemble des sommets est partitionné en clusters. Un des clusters ne contient que le dépôt. Chaque sommet est associé à une fenêtre de temps pendant laquelle la livraison doit avoir lieu si le sommet est visité. L'objectif est de trouver un circuit à coût minimum commençant et se terminant au dépôt, de sorte que chaque cluster soit visité une seule fois et que les contraintes de temps soient respectées, c'est-à-dire que, pour chaque cluster, un seul sommet est visité pendant sa fenêtre de temps. Dans ce chapitre, quatre programmes linéaires à variables mixtes pour le GTSPTW sont proposés et comparés. Les modèles diffèrent par la manière dont sont définies les variables de sélection des arcs et les variables temporelles: basées sur des sommets ou des clusters. Toutes les formulations sont compactes, ce qui signifie que le nombre de variables et de contraintes est polynomial par rapport à la taille de l'instance. Les relations de dominance entre les relaxations linéaires de ces formulations sont théoriquement établies. Nous avons également mené une étude expérimentale pour comparer la relaxation linéaire et les performances d'un algorithme de branch-and-bound pour les quatre formulations. Les résultats sur les relaxations linéaires montrent qu'en moyenne la formulation $\mathcal{F}1$ est la

meilleure, suivie de la formulation $\mathcal{F}2$. Toutefois, lors de la résolution du GTSPTW avec l'algorithme de branch-and-bound de CPLEX, la formulation $\mathcal{F}2$ est la plus efficace, en moyenne, suivie par la formulation $\mathcal{F}1$. Par conséquent, nous recommandons d'utiliser les formulations $\mathcal{F}1$ et $\mathcal{F}2$ pour la résolution du GTSPTW. De plus, des inégalités super-valides sont proposées pour les formulations $\mathcal{F}1$ et $\mathcal{F}2$, ce qui permet d'améliorer les performances.

Dans le chapitre 3, nous développons un algorithme de branch-and-cut pour le GTSPTW. Plusieurs familles d'inégalités valides sont proposées. Ces familles peuvent contenir un nombres de contraintes polynomial ou exponentiel. Ces familles de contraintes sont intégrées dans un algorithme de branch-and-cut via des procédures de séparation dédiées. L'algorithme de branch-and-cut comprend trois phases principales. La première phase est l'étape de pré-traitement visant à resserrer les fenêtres de temps de l'instance et à éliminer les arcs qui ne peuvent pas faire partie d'une solution réalisable. Ensuite, nous appliquons une heuristique efficace pour obtenir une solution réalisable et pour fournir une borne supérieure de la valeur optimale. Enfin, la phase principale consiste à résoudre le problème en utilisant un algorithme de branch-and-cut, basé sur le schéma standard de branch-and-cut fourni par le solveur commercial CPLEX 12.6.3. Le modèle initial est construit à partir du programme linéaire à variables mixtes $\mathcal{F}1$ ou $\mathcal{F}2$. La solution initiale obtenue par l'heuristique sert pour initialiser la procédure de branch-and-cut. Dans l'arbre de branch-and-bound, chaque fois qu'une solution fractionnaire est obtenue, les inégalités valides proposées sont vérifiées et celles violées par la solution courante sont ajoutées au modèle. Pour les inégalités valides avec un nombre polynomial de contraintes, nous les mémorisons toutes et parcourons l'ensemble pour rechercher toutes celles qui sont violées. Pour les familles de contraintes de taille exponentielle, des algorithmes de séparation sont appliqués pour détecter efficacement les inégalités violées et le nombre d'inégalités que nous avons choisi de séparer est limité. Nous testons l'algorithme sur trois groupes d'instances ayant des caractéristiques différentes. Les résultats démontrent clairement l'efficacité de l'algorithme de branch-and-cut proposé et la qualité de la formulation $\mathcal{F}2$. L'algorithme de branch-and-cut basé sur la formulation $\mathcal{F}2$ peut résoudre à l'optimum des instances d'environ 30 clusters en moins d'une heure de temps de calcul.

Enfin, dans le chapitre 4, nous étudions le cas avec plusieurs véhicules, appelé problème de tournées de véhicules généralisé avec fenêtres de temps (GVRPTW). Le GVRPTW est défini sur un graphe orienté $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ où l'ensemble des sommets $\mathcal{V}$ est partitionné en clusters. Un des clusters ne contient que le dépôt, où se trouve une flotte homogène de véhicules, chacun avec une capacité limitée. Les autres clusters

représentent les clients. Chaque cluster est associé à une demande. Dans un cluster, les sommets représentent les localisations possibles du client. Chaque sommet est associé à une fenêtre de temps pendant laquelle la visite doit avoir lieu si le sommet est visité. L'objectif est de trouver un ensemble de routes telles que le coût total de routage soit minimal, exactement un sommet par cluster soit visité, et toutes les contraintes de capacité et de temps soient respectées. Nous proposons une formulation en programme linéaire à variables mixtes et une formulation basée sur un problème de *set covering pour le GVRPTW. Sur la base de la formulation de* set covering, nous développons une heuristique basée sur la génération de colonnes pour le GVRPTW. Cette heuristique combine plusieurs composants, notamment une heuristique de construction, une procédure d'optimisation d'une route, un algorithme de recherche locale, et la génération de routes de coût réduit négatif. Les résultats expérimentaux sur des instances de la littérature montrent que l'algorithme proposé est très efficace et que des solutions sont de très bonne qualité et peuvent être obtenues dans des temps de calcul très courts pour des instances comprenant jusqu'à 120 clusters.

Les sujets abordés dans cette thèse permettent de fournir quelques remarques quant à la gestion de ces nouveaux services de livraison du dernier kilomètre avec plusieurs options d'expédition. Cependant, plusieurs pistes de recherche restent ouvertes. Nous listons ci-dessous celles qui, à notre avis, peuvent être intéressantes à développer.

La première perspective est du point de vue méthodologique. Pour le GTSPTW, la procédure que nous avons développée pour obtenir une solution initiale est déjà efficace mais pourrait être améliorée pour obtenir des solutions de très grande qualité. Les mouvements de recherche locale ou les méta-heuristiques classiques pour les problèmes de routage pourraient être adaptés au GTSPTW. Pour l'heuristique basée sur la génération de colonnes que nous avons proposée pour GVRPTW, les opérateurs de recherche locale sont limités. Des opérateurs plus sophistiqués dédiés au GVRPTW pourraient être proposés. La génération de colonne est appliquée de manière heuristique et fournit de très bonnes bornes supérieures. Il semble intéressant de travailler sur le sous-problème et le calcul d'une borne inférieure.

Dans cette thèse, nous avons supposé que les clients fournissaient plusieurs lieux de livraison possibles avec le même désir de recevoir le colis : les clients n'ont pas de préférence pour un lieu ou une période spécifique. Cependant, en réalité, les clients peuvent avoir des préférences différentes en ce qui concerne les lieux de livraison ou les périodes de livraison qu'ils indiquent. Pour prendre en compte cet aspect, nous pourrions associer chaque lieu de livraison à un facteur de satisfaction (profit). Ensuite, le nouveau problème pourrait être modélisé comme une variante du VRP avec profits

lorsque les clients sont associés à plusieurs localisation ou, de la même manière, comme une variante du GVRP où les localisations sont associées à un profit. Le facteur de satisfaction (profit) peut apparaître dans la fonction objectif ou dans les contraintes. Ce problème est intéressant car de plus en plus d'entreprises se concentrent sur la satisfaction des clients afin de pouvoir les fidéliser.

Dans cette thèse, les clients sont associés à plusieurs lieux de livraison pouvant représenter des consignes. Une consigne possède une capacité limitée que nous n'avons pas prise en compte. De ce fait, il est intéressant de travailler sur des problèmes de tournées de véhicules qui intègrent la gestion de la capacité associée aux emplacements de livraison. De plus, les colis livrés dans une consigne pourraient ne pas être récupérés par le client au cours de la même période de planification. Alors, la capacité de la consigne sera réduite tant que le colis n'aura pas été récupérés par le client. Par conséquent, des contraintes supplémentaires doivent être prises en compte dans les modèles et algorithmes lorsque la livraison dans des consignes est considérée.

Une autre perspective de ce travail consiste à examiner la version dynamique des problèmes de tournées de véhicules dans le contexte que la livraison du dernier kilomètre avec plusieurs options d'expédition. Lorsque certains lieux ou certaines fenêtres de temps associées à un client changent au cours de l'horizon de planification, une nouvelle solution doit être recalculée. L'heuristique basée sur la génération de colonnes que nous proposons pour le GVRPTW pourrait être utilisée dans de tels cas, car elle implique plusieurs composants pour construire ou optimiser des solutions, et les temps de calcul sont courts.

Models and Algorithms for Last Mile Delivery Problems with Multiple Shipping Options

Abstract

In this thesis, we study routing problems that arise in the context of last mile delivery when multiple delivery options are proposed to the customers. The most common option to deliver packages is home/workplace delivery. Besides, the delivery can be made to pick-up points such as dedicated lockers or stores. In recent years, a new concept called trunk/in-car delivery has been proposed. Here, customers' packages can be delivered to the trunks of cars. Our goal is to model and develop efficient solution approaches for routing problems in this context, in which each customer can have multiple shipping locations. First, we survey non-Hamiltonian routing problems. Then, we study the single-vehicle case in the considered context, which is modeled as a Generalized Traveling Salesman Problem with Time Windows (GTSPTW). Four mixed integer linear programming formulations and an efficient branch-and-cut algorithm are proposed. Finally, we study the multi-vehicle case which is denoted Generalized Vehicle Routing Problem with Time Windows (GVRPTW). An efficient column generation based heuristic is proposed to solve it.

Keywords: last mile delivery; trunk/in-car delivery; generalized traveling salesman problem; generalized vehicle routing problem; time windows; branch-and-cut.

Résumé

Dans cette thèse, nous étudions les problèmes de tournées de véhicules dans le contexte de la livraison du dernier kilomètre lorsque plusieurs options de livraisons sont proposées aux clients. Le mode de livraison le plus commun est la livraison à domicile ou au travail. La livraison peut également être effectuée dans des points de collecte tels que des consignes ou des magasins. Ces dernières années, un nouveau concept appelé livraison dans le coffre / dans la voiture a été proposé. Avec ce mode de livraison, les colis des clients peuvent être livrés directement dans les coffres des voitures. Notre objectif est de modéliser et de développer des approches de résolution efficaces pour les problèmes de routage dans ce contexte, dans lequel chaque client peut disposer de plusieurs lieux potentiels de livraison. Premièrement, nous proposons un état de l'art sur les problèmes de routage non-Hamiltoniens. Ensuite, nous étudions le cas avec un seul véhicule, qui est modélisé comme un problème du voyageur de commerce généralisé avec fenêtres de temps (GTSPTW). Quatre formulations en programme linéaire à variables mixtes et un algorithme efficace de branch-and-cut sont proposés. Enfin, nous étudions le cas multi-véhicules, dénommé problème de tournées de véhicules généralisé avec fenêtres de temps (GVRPTW). Une heuristique efficace basée sur la génération de colonnes est proposée pour le résoudre.

Mots clés : livraison du dernier kilomètre; livraison dans le coffre / dans la voiture; problème du voyageur de commerce généralisé; problème de tournées de véhicules généralisé; fenêtres de temps; branch-and-cut.
