UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dottorato di Ricerca in
Ingegneria dell'Innovazione Industriale

XXVIII Ciclo

Aggregated Formulations, Exact and Heuristic Algorithms for Pickup and Delivery Routing Problems

Bruno Petrato Bruck

Il Coordinatore
Prof. Mauro Dell'Amico

Il Tutor
Prof. Manuel Iori

A.A. 2013–2016

Contents

Acknowledgments

First of all, many thanks to my tutor Manuel Iori, who during these last 3 years of Ph.D. was such an excellent tutor, teaching me so much about Operations Research and helping me achieving this realization. Thanks also for all the opportunities you gave me and all the valuable suggestions, advice and encouragement, I really appreciate.

I would like to thank all the components of the group of Operations Research of the *Dipartimento di Scienze e Metodi dell'Ingegneria* of the University of Modena and Reggio Emilia, for the support and, in particular, also many thanks to Raphael Kramer and Stefano Novellani for the friendship.

Many thanks go also to my co-authors, that helped me in writing the reports that form part of this thesis: Prof. Jean-François-Cordeau from HEC Montréal, Prof. Anand Subramanian from the Federal University of Paraíba and Prof. Matteo Vignoli from the University of Modena and Reggio Emilia. Thanks also to the undergraduate, master and Ph.D. students I collaborated with during these years, in particular to Dario Vezzali, Jair Paulino, Fábio Cruz and Valerio Incerti.

I would like also to thank Tommaso Poncemi, Luca Baracchi and Diego Lancellotti for the opportunity to work with real world problems faced by their companies.

Many thanks also to all my friends who somehow helped me during this journey, in particular I would like to thank my friend Rodrigo for all the support and for the valuable friendship.

Finally, but not least importantly, I would like to thank my mother for being such an amazing person and for always having supported and encouraged me during my life.

Reggio Emilia, March 1, 2016

Bruno Petrato Bruck

List of Figures

List of Tables

Chapter 1

Introduction

The family of *vehicle routing problems* (VRPs) is perhaps one of the most studied topics in *combinatorial optimization* due to its high practical applicability and economic relevance. It involves finding a set of routes for a fleet of vehicles in which a series of transportation requests are performed at minimum cost. In particular, we must decide which vehicle handles which requests and in which sequence, so that all routes can be feasibly executed (Toth and Vigo 2014). One of the most important classes of vehicle routing problems is the so-called *pickup and delivery problems* (PDPs), where objects or people have to be transported from given origins to given destinations. These problems have been extensively studied in the last three decades and are not only theoretically sound but also have several practical applications. For a more in depth overview on these problems and on other several variants that have been addressed in the literature, we refer the interested reader to the surveys of Parragh et al. (2008a,b).

Berbeglia et al. (2007) propose a general framework that classifies PDPs into three main categories according to the structure of the routes and the type of customer demands. In the so-called *many-to-many* (M-M) problems, commodities may have multiple origins and destinations, and any location may serve as origin or destination for commodities. These problems have applications in many practical contexts, for instance, the repositioning of inventory between retail stores and in bike and car sharing systems. In the second category, namely *one-to-many-to-one* (1-M-1), some commodities originated from a depot must be delivered, while others have to be collected and brought back to the depot. Applications in this case arise mainly in the context of reverse logistics, for example, in the distribution of beverages and courier service transportation. Another interesting application refers to the distribution in the European electrical and electronic market, where there is a directive imposing that the delivery of new appliances should be accompanied by the pickup of old ones for proper disposal. Finally, in *one-to-one* (1-1) problems, each commodity is associated with a single origin and destination. These problems are typically found in urban courier operations, passenger and less-than-truckload transportation. Recent surveys treating PDPs

for transportation of goods and people have been presented by, respectively, Battarra et al. (2014) and Doerner and Salazar-González (2014).

The aim of this Ph.D. thesis is to considerably advance the state of the art in exact and heuristic algorithms applied to vehicle routing problems, and to study their application in a few important real life scenarios. We apply typical techniques of combinatorial optimization including integer linear programming, branch-and-bound and branch-and-cut algorithms, Benders decomposition, local search, matheuristics and metaheuristics to develop efficient and successful approaches to solve several VRPs and PDPs.

In Chapter 2 we turn our attention to *single vehicle 1-M-1* problems, common in the context of reverse logistics, for example, in the distribution of beverage, courier service transportation, in mail services and in the servicing of offshore platforms (see, e.g., Gribkovskaia and Laporte 2008). We base our studies on one of the most studied problems of this class, known as the *single vehicle routing problem with deliveries and selective pickups* (SVRPDSP). In this problem deliveries are mandatory but pickups are optional and generate a revenue if performed, and customers requiring both a delivery and a pickup (combined demand) can be visited either once or twice.

Most exact algorithms in the literature solve the SVRPDSP by looking for Elementary tours on an extended network, obtained by transforming each combined demand customer into two different customers, one requiring only the delivery and the other one only the pickup. Because this can result in a significant loss in performance, we focus instead on the original problem network and present formulations that can yield non-Elementary tours. Through the use of Benders Decomposition, valid inequalities, and tailored optimization techniques based on branch-and-cut frameworks, we develop exact algorithms that outmatch previous results in the literature and obtain proven optimal solutions for all benchmark instances. We then generalize the algorithms to solve several other PDPs, including the cases of split deliveries, intermediate dropoffs, mandatory pickups, and multiple vehicles.

Chapter 3 focus instead on a *single vehicle M-M* problem called the *static bike sharing rebalancing problem without temporary operations* (SBRPW). As the name indicates, we concentrate on the application of this problem in the context of bike sharing systems, which is a fast growing market with hundreds of initiatives spread all over the world. These systems are an important tool used by many governments to support sustainable mobility, reduce urban traffic and pollution. Notice that the application of the SBRPW is not restricted to bike sharing systems and appears in other practical contexts, such as in the repositioning of inventory between retail stores, courier service transportation and the management of car sharing systems. In the SBRPW there is a depot and a set of stations requiring a certain demand of bikes to be either delivered or picked up. Some stations might be initially balanced and may not be visited. Then, a single vehicle is used to perform a series of deliveries and pickups to bring each station to a balanced state. Multiple visits and split of demand is al-

lowed at unbalanced stations, however it is strictly forbidden to use stations as buffers, either by performing temporary deliveries or temporary pickups. To our knowledge, this is the first work to directly approach the SBRPW. In this chapter, we formally introduce the problem, present a few interesting theoretical results, propose two exact algorithms both based on an efficient feasibility test algorithm. Our algorithms are then tested on a set of benchmark instances derived from the related literature and extensive computational results are shown.

Chapters 2 and 3 focused on PDPs, while on Chapter 4 we focus instead on a practical VRP variant. In recent years, the increasing awareness of governments, as well as companies and individuals, of the damages to the environment caused by human activities have stimulated the study of vehicle routing problems applied to this context. In this sense, the reduction of CO_2 emissions is an important topic that is pursued through a range of practices. A relevant example is carpooling, which is defined as the act of individuals sharing a single car. In Chapter 4 we study a 1-1 practical daily carpooling problem found in a large Italian service company, Coopservice Soc.coop.p.A. with headquarters in Reggio Emilia (Italy). Our objective is to develop an integrated web application to be used by the employees of this company to organize carpooling crews on a daily basis, so as to reach a common destination. We look for possible crews by the use of two mathematical formulations and two heuristic algorithms. The heuristic algorithms are then embedded into the web application to provide users with carpooling solutions.

In Chapter 5 we approach a practical attended home delivery time slot management problem found in a large Italian company, which offers gas and water services to millions of habitants in the region of Emilia Romagna (Italy). Due to a series of laws established by the European Union, the company was forced by the Italian authority on the sector to provide customers with an online application, where they are able to make requests for services by booking time slots. Because the company acts on a large area, they divided their territory into 12 regions, each having its own time slot table. For each time table the company has to decide which and how many time slots to offer considering the expected demand of the region, a minimum level of service and the expected service costs. Because the actual costs can only be evaluated once the demand is known, the problem becomes fairly complex as it involves three interconnected levels. The first level refers to the design of the time tables themselves, while the second level is the actual choice of time slots by customers, consequently revealing the demand. Then, in the third level a routing plan is created to minimize the total distance travelled and final costs are estimated. To approach this problem we propose a *large neighborhood search* algorithm, which uses as a repair method an *integer linear programming* formulation to bring solutions to a feasible state. To evaluate the expected cost of solutions we propose an algorithm that simulates customers choices using one of a few simulation strategies and then exactly solves a routing problem to generate the routes for each operator.

Although this thesis is dedicated to the study of vehicle routing problems, in Appendix

A we study a practical problem in the context of *group decision making*, found in an Italian company called H2H Facility Solutions, that acts on the sector of *facility management*. We present a methodology that aims at developing a *decision support system* for the selection of the most suitable supplier to provide services to clients. We performed interviews with employees at the company and identified the main criteria used by decision makers to choose suppliers. Then, by applying the well known *analytic hierarchy process* (AHP) method (see, e.g., Ishizaka and Labib 2011) we derive weights for each criterion. Additionally, we use a method called the *revised Simos' procedure* to derive a second set of weights for the same criteria, which are compared with the ones generated by the AHP. This allows us to have insights on the consistency of the weights and use this information to further refine them. We present the results of both methods and future research directions.

Notice that the chapters of this thesis are self-contained entities as they refer to independent projects and separate papers. The advantage of this organization is that it provides a logical and fluid view of the evolution of the conducted research.

1.1 Bibliography

M. Battarra, J.-F Cordeau, and M. Iori. Pickup-and-delivery problems for goods transportation. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, pages 161–191. SIAM, 2nd edition, 2014.

G. Berbeglia, J.-F Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: aclassification scheme and survey. *Top*, 15(1):1–31, 2007.

K.F. Doerner and J.-J. Salazar-González. Pickup-and-delivery problems for people transportation. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, pages 193–212. SIAM, 2nd edition, 2014.

I. Gribkovskaia and G. Laporte. One-to-many-to-one single vehicle pickup and delivery problems. In B. Golden, S. Raghavan, E. Wasil, R. Sharda, and S. Voss, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 359–377. Springer, Berlin, 2008.

A. Ishizaka and A. Labib. Review of the main developments in the analytic hierarchy process. *Expert Systems with Applications*, 38(11):14336–14345, 2011.

S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58 (1):21–51, 2008a.

S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008b.

P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. SIAM, 2nd edition, 2014.

Chapter 2

Non-elementary formulations for single vehicle routing problems with pickups and deliveries

In this chapter we study the class of one-to-many-to-one single vehicle routing problems with pickups and deliveries, in which a single capacitated vehicle is used to serve a set of customers requiring a delivery, a pickup, or both. These problems appear in many real-world contexts, including beverage distribution, courier service transportation, and reverse logistics. We first focus on the most studied problem in this class, known as the *single vehicle routing problem with deliveries and selective pickups*, in which deliveries are mandatory but pickups are optional and generate a revenue if performed. By the use of Benders Decomposition, valid inequalities, and tailored optimization techniques based on branch-and-cut frameworks, we develop exact algorithms that outmatch previous results in the literature and that obtain proven optimal solutions for all benchmark instances. These algorithms are then generalized to solve several other vehicle routing problems with pickups and deliveries, including the cases of split deliveries, intermediate dropoffs, mandatory pickups, and multiple vehicles.

2.1 Introduction

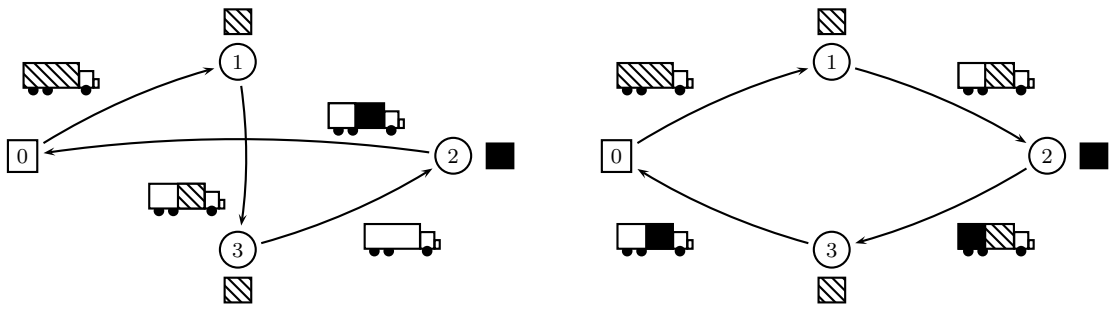
We address a large class of *pickup and delivery* routing problems, where a single vehicle leaves the depot to perform both a series of deliveries of a first commodity and a series of collections of a second commodity which is brought back to the depot. These problems are known as *one-to-many-to-one single vehicle pickup and delivery problems* (1-M-1 SVPDPs), see Berbeglia et al. (2007) and Gribkovskaia and Laporte (2008), and are among the most important problems in the large area of pickup and delivery because they can model many real-world situations. The classical example found in textbooks arises in *beverage distribution*,

and is that of a capacitated vehicle that delivers full drink bottles to customers and retrieves empty bottles to the depot (see, e.g., Golden et al. 2002). Several other interesting applications can be found, among which we name the distribution in the European *electrical and electronic market*, where the WEEE directive imposes that the delivery of new appliances should be accompanied by the pickup of old ones, *courier service transportation*, where the courier delivers packages to customers and brings back to the depot other packages to be shipped later on, and in general the large area of *reverse logistics*, whose aim is to minimize the kilometers traveled with an empty load (see, e.g., Daniel et al. 2009).

To better model situations arising in various real-world contexts, different problem variants have been studied. The relevant literature has focused on two classes of problems:

- *single demand* (SD): problem instances contain only customers that require a delivery or a pickup, but not both;
- *combined demand* (CD): problem instances may contain any type of customers.

Figure 2.1 shows two important SD cases. The delivery commodity is depicted with dashed lines and the pickup commodity in black. Vertex 0 represents the depot, vertices 1 and 3 are *delivery customers* (also referred to as *linehaul customers* in the literature), and vertex 2 is a *pickup customer* (also known as *backhaul customer*). Figure 2.1-(a) depicts the *SVPDP with backhauls*, in which all deliveries are performed before any pickup is made. In Figure 2.1-(b) the backhaul constraint is relaxed and *mixed deliveries* are allowed, thus pickup and delivery operations can be performed in any order. This problem, called *mixed SVPDP*, is less constrained than the former. Its solution could possibly require reshuffling of the cargo, but usually consists of a shorter route.



(a) backhauls (b) mixed deliveries

Figure 2.1: *Two common 1-M-1 SVPDPs single demand variants.*

Regarding the CD case, if only a *single visit* is allowed to any customer, then the problem can be transformed into an SD case by an easy modification of the input (see again Gribkovskaia and Laporte 2008). Variants where *multiple visits* are allowed have been studied because they can lead to significant cost reductions. Figure 2.2 presents three common

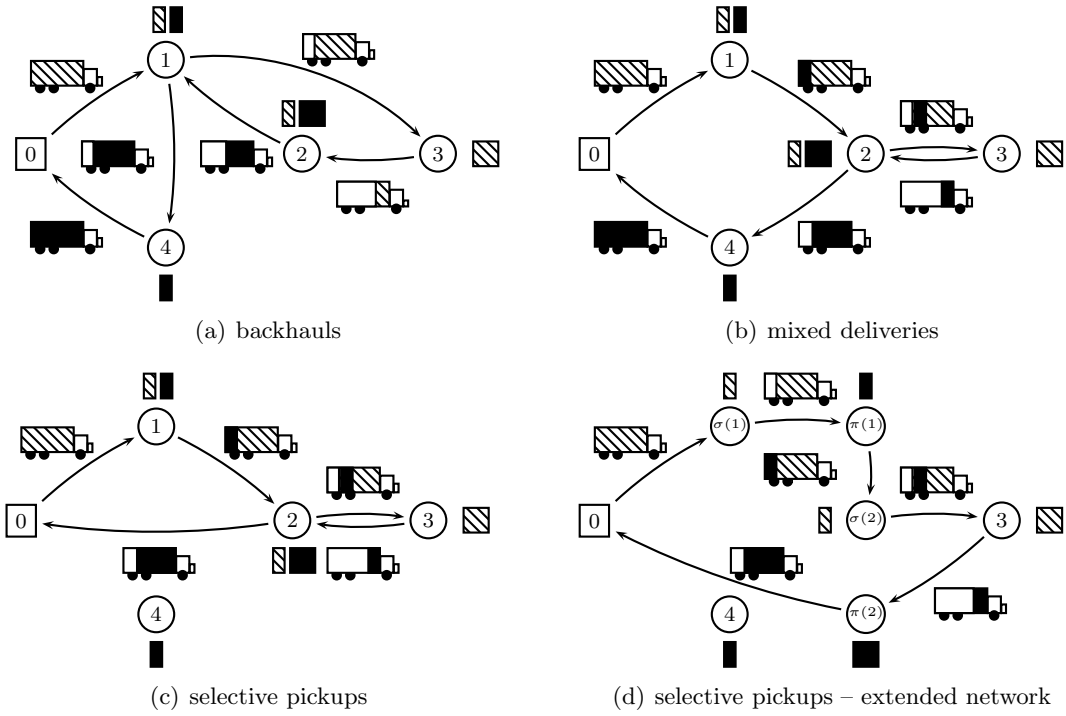
variants with multiple visits (disregard for the moment Figure 2.2-(d)). Customer 3 requires only a delivery and 4 only a pickup, whereas the *combined demand customers* (*combined customers* for short in the following) 1 and 2 are characterized by both a delivery and a pickup. In Figure 2.2-(a) the vehicle adopts backhaul deliveries, by first performing all deliveries (1, 3, and 2), and then all pickups (2, 1, and 4). This problem is an extension to the CD case of the problem in Figure 2.1-(a). Figure 2.2-(b) depicts an example with mixed deliveries, where the vehicle performs delivery and pickup in 1, delivery in 2, delivery in 3, pickup in 2 (not possible during the first visit because of the capacity constraint), and pickup in 4. The problem is known as *single vehicle routing problem with pickups and deliveries* (and also as *general SVPDP*) and extends the one in Figure 2.1-(b). In all the cases discussed so far the pickup operations are *mandatory*. Figure 2.2-(c) depicts instead the case of *selective pickups*, where not only the operations can be performed in mixed order, but also pickups are optional and give a revenue if performed. In the example the pickup in 4 is not performed because not profitable enough.

In the survey by Gribkovskaia and Laporte (2008) the problem depicted in Figure 2.2-(c) is denoted as the *single vehicle routing problem with deliveries and selective pickups* (SVRPDSP). The SVRPDSP is probably the most studied problem among the 1-M-1 SVPDPs. The first mention of it that we know of is in Golden and Assad (1986), that suggest that the study of this distribution strategy is of interest because it can lead to a lower transportation cost.

It is easy to see that the SD and CD versions of the SVRPDSP are interchangeable. On one hand, the CD case is clearly a generalization of the SD case. On the other hand, combined demands can be transformed into single demands by *duplicating* each combined customer into two customers: a delivery and a pickup. In the following we will refer to the network obtained by this duplication as to the *extended network*. An example of such a network is depicted in Figure 2.2-(d) and is obtained by duplicating the combined customers of Figure 2.2-(c).

As far as we know, most of the literature on the SVRPDSP made use of this duplication (that will be formally described in Section 2.2) and then focused on the solution of the SD case. Some exceptions, discussed in details in Section 2.2.1, are provided by Gribkovskaia et al. (2008) and Nagy and Salhi (2005). The rationale for using the extended network is that in any optimal solution to an SD instance no customer needs to be visited twice, and thus known results from famous problems, including the *traveling salesman problem* (TSP), can be reused. For simplicity, in the following we call this type of solutions *Elementary*. This duplication comes however at a price, because the number of vertices can double, thus obliging the algorithms to deal with much larger problem networks.

In this chapter we develop exact algorithms for the SVRPDSP and then show how to adapt them to other 1-M-1 SVPDPs, including problems with mandatory pickups. We focus on the CD case, but instead of making use of the extended network, we work on the original



(a) backhauls (b) mixed deliveries

(c) selective pickups (d) selective pickups – extended network

Figure 2.2: *Some common 1-M-1 SVPDPs combined demand variants.*

problem network. We thus deal with solutions that can be *non-Elementary*, where typically a few customers are visited twice. Consequently, the mathematical models that we developed contain variables that are not restrained to be binary and do not admit classical results from the TSP literature. To deal with this situation we developed innovative algorithms. Computational results show that the developed techniques concretely advance the state-of-the-art on the SVRPDSP and that are able to solve to optimality all known benchmark instances (in all the single vehicle problem variants that we addressed) in reasonable computational times.

In many real-world scenarios the distribution process is performed by a fleet of vehicles instead of a single one. The models and algorithms that we implemented for the single vehicle case can be used to obtain interesting insights also on this larger class of problems. In particular, in the last part of this chapter we show how our best algorithm can be quite easily adapted to consider multiple homogeneous capacitated vehicles, and we present extensive results on a new set of instances.

The remainder of the chapter is organized as follows. In Section 2.2 a formal description of the SVRPDSP is given and the related literature is presented. In Section 2.3 a standard formulation looking for Elementary tours in the extended network and based on the use of two-commodity flow variables is introduced. In Section 2.4 we present a more innovative non-Elementary formulation, and show that it provides an optimal solution to the SD case and a valid relaxation to the CD case. In Section 2.5 we improve the non-Elementary formulation

by means of Benders decomposition and cutting planes. In Section 2.6 we embed the proposed techniques into a list of exact branch-and-cut algorithms, that are then computationally tested in Section 2.7 on several problem variants.

2.1.1 Main Contributions

The main contributions of our work are the following ones:

- we propose a non-Elementary formulation making use of two-commodity flow variables and solving the SVRPDSP variant in which split deliveries and intermediate dropoffs are allowed;
- we prove that the formulation provides the optimal solution for the SD case and a tight lower bound for the CD case;
- we obtain an enhanced formulation by a Benders decomposition that projects out the two-commodity flow variables; we further improve the formulation by proposing new families of valid inequalities and exact polynomial time separation procedures;
- we embed everything into exact branch-and-cut algorithms and obtain a substantial breakthrough in the computational results for the SVRPDSP: existing algorithms solved to proven optimality only instances with up to 22 combined demand customers (see Gribkovskaia et al. 2008) or 68 single demand customers (see Gutiérrez-Jarpa et al. 2009), whereas our best algorithm solves to optimality with comparable times all benchmarks with up to 100 combined demand customers;
- we provide insights that can be useful for practitioners, showing how the main features of the SVRPDSP can lead to relevant cost savings;
- we adapt the above techniques to deal with a wide range of 1-M-1 pickup and delivery problems, including the cases of mandatory pickups, split deliveries, intermediate dropoffs, and distribution by multiple vehicles.

2.2 Problem Description

In the *single vehicle routing problem with deliveries and selective pickups* (SVRPDSP) we are given a complete directed graph $G = \{V, A\}$, where $V = \{0, 1, 2, \ldots, n\}$, vertex 0 is the depot, vertices $1, \ldots, n$ are the customers, and $A = \{(i, j) : i, j \in V, i \neq j\}$. For convenience of notation we partition the customer set into three subsets, namely P, D, and PD. Each delivery customer $i \in D$ requires a delivery of weight d_i, each pickup customer $i \in P$ offers a pickup of weight p_i and revenue r_i, whereas each combined customer $i \in PD$ is associated with both a delivery of weight d_i and a pickup of weight p_i and revenue r_i.

Each delivery customer must be visited exactly once to perform the delivery, but visits to pickup customers are optional. Each combined customer can be visited either once or twice: if visited twice, then its delivery and pickup are performed separately; if visited once, then either only its delivery is performed, or its delivery and pickup are performed simultaneously. In the following we refer to G as to the *original network* of the SVRPDSP.

A single vehicle of weight capacity Q, with $Q \geq \sum_{i \in V} d_i$, is based at the depot. A traveling cost c_{ij} is associated with each arc $(i, j) \in A$. We consider the general case in which the cost matrix may be asymmetric and may not satisfy the triangular inequality. The SVRPDSP is to find a route that starts and ends at the depot, performs all deliveries and possibly some pickups, and minimizes the total cost given by the traveling cost minus the revenues generated by the collected pickups. The problem is strongly NP-hard because generalizes the asymmetric version of TSP, arising as a special SVRPDSP case when $P = PD = \emptyset$.

2.2.1 Literature Review

The first mention to the SVRPDSP that we know of is by Golden and Assad (1986), that suggest it as a natural extension of the standard backhaul problem that allows to take full advantage of backhaul economies. Since then, several algorithms and models have been implemented for solving the SVRPDSP. Most of these attempts are based on Elementary solutions.

Süral and Bookbinder (2003) focused on the SD case (i.e., $PD = \emptyset$) and suggested the use of the extended network that we discussed in Section 2.1 to deal with the CD case. They introduced a classification of the SD problems, and then studied the SVRPDSP (that they called *single vehicle routing problem with unrestricted backhauls*), motivated by the fact that it is the most general problem of their classification. They proposed a compact *mixed integer linear programming* (MILP) formulation based on the classical constraints by Miller et al. (1960), and enriched it with families of valid inequalities. Their algorithm failed in solving to optimality some instances with just 10 SD customers.

Gribkovskaia et al. (2008) studied the CD case, and focused on instances in which all customers require a delivery (i.e., $P = \emptyset$). They developed a new MILP formulation based on the original network, but obtained by doubling combined customers into two vertices: the first is associated with the first visit in which delivery or both delivery and pickup are performed, the second represents the second visit in which a pickup may be performed. They made use of classical vehicle flow variables among the newly created vertices, plus two additional sets of binary variables, indicating, respectively, whether pickup and delivery were performed simultaneously, and whether pickup was performed during the second visit. As for the case of the extended network, this strategy too comes at the price of requiring a large number of vehicle flow variables. Their MILP was improved with several valid inequalities, nevertheless, only instances with up to 22 CD customers (i.e., 44 SD customers) could

be solved to optimality within a reasonable time. In addition, they developed a tabu search heuristic, also working on the extended network, and computationally evaluated it on random instances, finding that the best solutions were frequently non-Elementary. Gutiérrez-Jarpa et al. (2009) addressed the symmetric version of the SVRPDSP. Similarly to Süral and Bookbinder (2003) they solved the single demand case and referred to the extended network to deal with combined demands. They proposed a MILP formulation only containing vehicle flow variables, and imposed subtour elimination and capacity restrictions by means of families of exponentially-many constraints. The resulting formulation was solved by a branch-and-cut algorithm that obtained favorable results, solving all instances with up to 68 SD customers.

Several papers addressed the SVRPDSP with heuristic algorithms. Here we mention only that the most effective algorithms are, according to our knowledge, the evolutionary algorithm by Bruck et al. (2012), and the variable neighborhood search by Coelho et al. (2012), both working on the extended network and hybridized by the use of MILP models. These heuristics could solve instances with up to 100 vertices, but obtained quite large optimality gap (See the supplementary material provided in Section 2.C). Heuristics working on the original problem network were developed by Gribkovskaia et al. (2008) for the SVRPDSP and by Nagy and Salhi (2005) for problems with mandatory pickups, and both made use of customer duplications only when necessary.

The use of selective pickups allows to model situations where the capacity of the vehicle is not enough to perform all operations. In some real-world applications concerning third-party logistics providers pickups can be simply skipped if they are not profitable enough. This is the case, for example, studied by Privé et al. (2006), in which "Distribution Jacques Dubois" delivers full bottles to customers in the Quebec City area and gets a revenue for bringing empty bottles back to the distribution center. In other applications pickups will eventually have to be made. In this case optimized solutions may be obtained by executing the SVRPDSP in a *rolling horizon* fashion, see, e.g., Cordeau et al. (2015), possibly increasing the revenues of those pickups that are closer to a certain deadline or have waited too much.

A complete review on the large literature on pickup and delivery routing problems is out of the scope of this chapter. For the 1-M-1 SVPDPs we refer the interested reader to the classification and survey given by Gribkovskaia and Laporte (2008). For general pickup and delivery routing problems, we refer to the recent surveys by Battarra et al. (2014) and Doerner and Salazar-González (2014), the former on the transportation of goods and the latter on the transportation of people.

2.3 An Elementary Formulation on the Extended Network

Let us first formally define the extended network as follows.

Definition 1 The *duplication* of a combined customer $i \in PD$ consists in replacing i by

two customers, a delivery customer $\sigma(i)$ and a pickup customer $\pi(i)$, having $d_{\sigma(i)} = d_i$, $p_{\sigma(i)} = 0$, $r_{\sigma(i)} = 0$, $d_{\pi(i)} = 0$, $p_{\pi(i)} = p_i$, $r_{\pi(i)} = r_i$, and costs $c_{\sigma(i),j} = c_{\pi(i),j} = c_{ij}$ and $c_{j,\sigma(i)} = c_{j,\pi(i)} = c_{ji}$ for all $j \in V$.

DEFINITION 2 Given an SVRPDSP instance on a graph $G = (V, A)$, the *extended network* is the graph $G' = \{V', A'\}$ obtained by applying the duplication of Definition 1 to each combined customer $i \in PD$, so obtaining $D' = D \cup \{\sigma(i) : i \in PD\}$, $P' = P \cup \{\pi(i) : i \in PD\}$, $V' = \{0\} \cup P' \cup D'$, and $A' = \{(i, j) : i, j \in V', i \neq j\}$.

On the basis of Definition 2, we can model the SVRPDSP using the following formulation looking for Elementary solutions on the extended network. Let x_{ij} be a binary variable indicating whether arc $(i, j) \in A'$ is used or not, and y_j be an additional binary variable taking value 1 if the pickup of customer $j \in P'$ is performed, 0 otherwise. Let also f^d_{ij} and f^p_{ij} be non-negative variables giving, respectively, the total delivery and pickup quantities transported by the vehicle when traveling along arc $(i, j) \in A'$. The resulting *two-commodity Elementary extended* (TCEE) formulation is then:

$$\text{(TCEE)} \quad \min\ z_{TCEE} = \sum_{(i,j)\in A'} c_{ij}x_{ij} + \sum_{j\in P'} r_j(1 - y_j) \tag{2.1}$$

subject to

$$\sum_{i\in V'\setminus\{j\}} x_{ij} = 1 \qquad \forall\ j \in D' \cup \{0\}, \tag{2.2}$$

$$\sum_{i\in V'\setminus\{j\}} x_{ij} = y_j \qquad \forall\ j \in P', \tag{2.3}$$

$$\sum_{i\in V'\setminus\{j\}} (x_{ij} - x_{ji}) = 0 \qquad \forall\ j \in V', \tag{2.4}$$

$$f^d_{ij} + f^p_{ij} \leq Qx_{ij} \qquad \forall\ (i, j) \in A', \tag{2.5}$$

$$\sum_{i\in V'\setminus\{j\}} \left(f^d_{ij} - f^d_{ji}\right) = d_j \qquad \forall\ j \in V' \setminus \{0\}, \tag{2.6}$$

$$\sum_{i\in V'\setminus\{j\}} \left(f^p_{ji} - f^p_{ij}\right) = p_j y_j \qquad \forall\ j \in P', \tag{2.7}$$

$$\sum_{i\in V'\setminus\{j\}} \left(f^p_{ji} - f^p_{ij}\right) = 0 \qquad \forall\ j \in D', \tag{2.8}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\ (i, j) \in A', \tag{2.9}$$

$$y_j \in \{0, 1\} \qquad \forall\ j \in P', \tag{2.10}$$

$$f^d_{ij}, f^p_{ij} \geq 0 \qquad \forall\ (i, j) \in A'. \tag{2.11}$$

Constraints (2.2) impose that each delivery vertex is visited once. Constraints (2.3) tie together the x with the y variables, whereas (2.5) connect the x with the f^d and f^p variables and ensure that the vehicle capacity is not exceeded. Constraints (2.4) and (2.6) impose flow conservation for the vehicle and the delivery quantity, respectively. Similarly, constraints (2.7) and (2.8) impose flow conservation of the pickup quantity for the vertices in P' and D',

respectively.

The objective function (2.1) minimizes the sum of the costs plus the sum of the lost revenues of those pickups that were not performed. We found it convenient to adopt this function because it always takes non-negative values. Minimizing (2.1) is equivalent to minimizing the original SVRPDSP objective (traveling cost minus revenues generated by the collected pickups), because the two functions just differ by the fixed term $\sum_{j\in P'} r_j$. Moreover (2.1) is widely used in the related *traveling salesman problem with profits* (TSP-P), also known as the prize-collecting traveling salesman problem, see, e.g., Feillet et al. (2001). Also note that one could impose initial values for the pickup and delivery quantities leaving the depot, by setting $\sum_{i\in V'} f^d_{0i} = \sum_{i\in D'} d_i$ and $\sum_{i\in V'} f^p_{0i} = 0$. Alternatively, formulation TCEE may return an excess of these quantities (for instances having loose capacity constraint), but these can be easily removed by inspection.

2.4 A Relaxation Based on a Non-Elementary Formulation

Let us consider the original SVRPDSP network $G = \{V, A\}$ described in Section 2.2. Similarly to what was done for formulation TCEE, let y_j be a binary variable taking value 1 if the pickup of customer $i \in P \cup PD$ is performed, 0 otherwise, and let f^d_{ij} and f^p_{ij} be non-negative variables giving, respectively, total delivery and pickup quantities transported along arc $(i, j) \in A$. Let also x_{ij} be an integer variable indicating the *number of times* in which arc $(i, j) \in A$ has been traveled. For feasibility, x_{ij} can take value 2 only when both i and j are in PD, but cannot be greater than 1 otherwise. By defining $A(PD) = \{(i, j) \in A : i, j \in PD\}$, we obtain the following *two-commodity non-Elementary* (TCNE) formulation:

$$\text{(TCNE)} \quad \min\ z_{TCNE} = \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in P\cup PD} r_j(1 - y_j) \tag{2.12}$$

subject to

$$\sum_{i\in V\setminus\{j\}} x_{ij} = 1 \qquad \forall\ j \in D \cup \{0\}, \tag{2.13}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij} = y_j \qquad \forall\ j \in P, \tag{2.14}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij} \geq 1 \qquad \forall\ j \in PD, \tag{2.15}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij} \leq y_j + 1 \qquad \forall\ j \in PD, \tag{2.16}$$

$$\sum_{i\in V\setminus\{j\}} (x_{ij} - x_{ji}) = 0 \qquad \forall\ j \in V, \tag{2.17}$$

$$f^d_{ij} + f^p_{ij} \leq Qx_{ij} \qquad \forall\ (i, j) \in A, \quad (t_{ij}) \tag{2.18}$$

$$\sum_{i\in V\setminus\{j\}} \left(f^d_{ij} - f^d_{ji}\right) = d_j \qquad \forall\ j \in V \setminus \{0\}, \quad (v_j) \tag{2.19}$$

$$\sum_{i \in V \setminus \{j\}} \left(f^p_{ji} - f^p_{ij}\right) = p_j y_j \qquad \forall\, j \in P \cup PD, \quad (w_j) \qquad (2.20)$$

$$\sum_{i \in V \setminus \{j\}} \left(f^p_{ji} - f^p_{ij}\right) = 0 \qquad \forall\, j \in D, \quad (w_j) \qquad (2.21)$$

$$x_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \setminus A(PD), \qquad (2.22)$$

$$x_{ij} \in \{0,1,2\} \qquad \forall\, (i,j) \in A(PD), \qquad (2.23)$$

$$y_j \in \{0,1\} \qquad \forall\, j \in P \cup PD, \qquad (2.24)$$

$$f^d_{ij}, f^p_{ij} \geq 0 \qquad \forall\, (i,j) \in A. \qquad (2.25)$$

The objective function (2.12) minimizes the sum of the costs and of the lost revenues. Constraints (2.13) force the delivery vertices and the depot to be visited once, whereas (2.15) force the combined vertices to be visited at least once. Constraint (2.16) are necessary to impose one visit in case the pickup is not performed, or up to two visits in case the pickup is performed. Constraints (2.14) tie together the x with the y variables for the pickup vertices. Constraints (2.17) impose flow conservation for the vehicle. Note that in (2.17) the sum of the variables associated with the incoming arcs on j may take value 0, 1, or even 2 when $j \in PD$. Constraints (2.18) and (2.19) impose the vehicle capacity restriction and the flow conservation for the delivery quantities, respectively. Similarly, constraints (2.20) and (2.21) set the flow conservation for the pickup quantities. Next to constraints (2.18)-(2.21) we provide the dual variables that are used later in Section 2.5. Note that in (2.18) the total quantity traveling along an arc in $A(PD)$ may take value up to $2Q$. An example of a TCNE solution in which an arc is traveled twice is given in Section 2.B.1 of the supplementary material.

Formulation TCNE has some interesting properties.

Property 1 *For the SD case formulation TCNE is equivalent to formulation TCEE and thus provides the optimal SVRPDSP solution value.*

This can be trivially checked by removing in TCNE variables and constraints associated with PD.

Property 2 *For the CD case formulation TCNE provides a relaxation of the SVRPDSP.*

Proof Given in Section 2.A.1 of the supplementary material.

A consequence of Property 2 is that in the general CD case the optimal solution value of formulation TCNE is a valid lower bound on the optimal SVRPDSP objective value, i.e., $z_{TCNE} \leq z_{TCEE}$. To see that there can be cases in which this lower bound is not tight, we discuss two relevant situations in the next two sections.

2.4.1 Solutions with Split Deliveries or Pickups

Using a common definition in the routing literature, we say that a delivery is *split* if it is partially performed during the first visit to the customer and then completed during

the second visit. The same definition applies to pickups. A solution with *split deliveries or pickups* is a solution in which one or more deliveries and/or one or more pickups are split.

This is a characteristic that can be observed in optimal TCNE solutions to CD instances. Refer for example to Figure 2.3. The figure reports for each vertex j the delivery and pickup quantities in square brackets, i.e., $[d_j, p_j]$, and for each arc (i, j) the delivery and pickup flows in round brackets, i.e., (f^d_{ij}, f^p_{ij}). The vehicle capacity is $Q = 30$. Figure 2.3-(a) shows an example in which the vehicle leaves the depot and performs a first visit to vertex 1 delivering entirely the quantity $d_1 = 5$ and picking up only 5 units of the quantity $p_1 = 10$. The remaining 5 units are picked up in the second visit to the vertex, and the resulting distribution would not invalidate the TCNE constraints. Split deliveries or pickups are not a major issue,

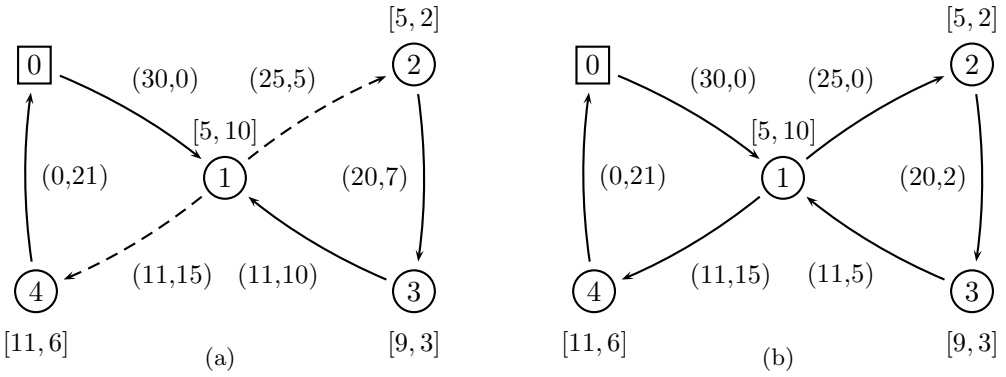


Figure 2.3: *Split deliveries or pickups*: (a) a feasible TCNE solution with a split pickup in vertex 1; (b) a same-cost solution without split that is feasible for the SVRPDSP.

because, as noted by a few authors in related 1-M-1 SVPDPs (see, e.g., Gribkovskaia and Laporte 2008), it is never suboptimal to perform the entire delivery first. Refer for example to Figure 2.3-(b), that depicts a solution obtained from that of Figure 2.3-(a) by modifying the loads on the arcs. During the first visit to vertex 1 the vehicle delivers the entire quantity d_1 but performs no pickup, and then during the second visit it performs the entire pickup p_1. This new solution has the same cost of the previous one, but it is now feasible for the SVRPDSP. Formally, we can state the following result.

PROPERTY 3 *A TCNE solution with split deliveries or pickups can be transformed into a solution having the same cost and for which no delivery or pickup is split.*

Proof Given in Section 2.A.2 of the supplementary material.

Given a solution with split deliveries or pickups, the flows f^d_{ij} and f^p_{ij} on the arcs that ensure to have a solution without splits may be found by making use of a simple recursive algorithm, called REMOVESPLIT, that we describe in details in Section 2.B.2 of the supplementary material. We just notice here that this algorithm has a complexity that grows exponentially with the number of vertices visited twice, because each such vertex creates two different paths.

2.4.2 Solutions with Intermediate Dropoffs

Using another common definition in the routing literature, we say that a *dropoff* at a vertex i occurs if the vehicle drops part of its load in i during its first visit, and recollects it during its second visit. An example is depicted in Figure 2.4, where the notation adopted is the same as the one in Figure 2.3 and the vehicle capacity is again 30. Figure 2.4-(a) shows an example in which the vehicle leaves the depot with 30 units of load and visits vertex 1, performing the delivery $d_1 = 5$ but also dropping off 2 units of load. This allows to perform completely the pickup at vertex 2, without exceeding the vehicle capacity on the following arc $(2, 3)$. The two units are then re-loaded on the vehicle during the second visit to 1.

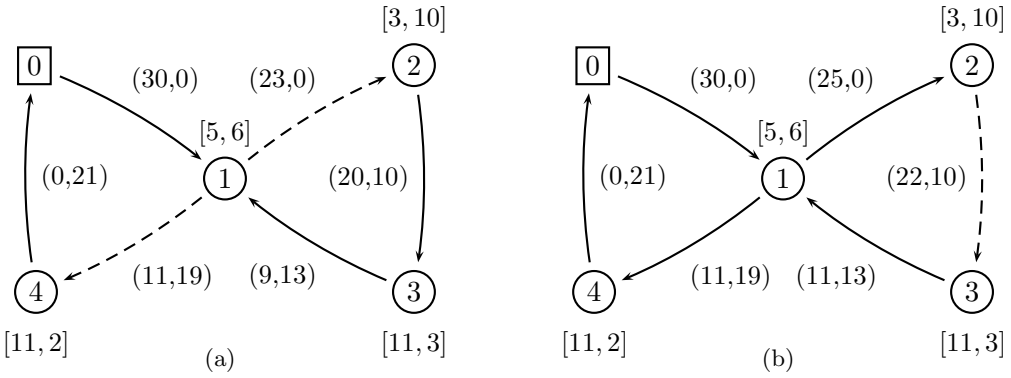


Figure 2.4: *Intermediate dropoffs*: (a) two units of load are temporarily dropped off at vertex 1; (b) the solution without dropoff violates capacity $Q = 30$ on arc (2,3).

For the case of dropoffs, unfortunately, there is no equivalent of the nice Property 3 that is valid for the case of split deliveries. The reason for this can be intuitively seen from Figure 2.4-(b): if the load is not dropped off at node 1 and the pickup of vertex 2 is still performed, then the capacity of the vehicle is exceeded on arc $(2, 3)$. We thus took care of the dropoffs by devising a few tailored algorithms, described in Section 2.6. These algorithms eventually need to check whether or not a solution has dropoffs. This is performed by a procedure called CHECKDROPOFF and presented in Section 2.B.3 of the supplementary material. This procedure is based on solving a modified version of formulation TCEE in which we include additional information derived from the Non-Elementary solution at hand.

We conclude this section by noticing the following fact.

PROPERTY 4 *Let the SVRPDSP-D be the relaxation of the SVRPDSP in which intermediate dropoffs of the load are allowed at the vertices. An optimal solution to the SVRPDSP-D may be obtained by solving formulation TCNE and then executing procedure* REMOVESPLIT *on the resulting solution.*

We believe this statement is of some interest, because, as noted by Gribkovskaia and Laporte (2008), intermediate drops have never been studied in the area of 1-M-1 SVPDPs,

so, as far as we know, Section 2.7 presents the first computational results for this research area. Note that the SVRPDSP-D does not allow split deliveries, and that is why REMOVESPLIT is needed in Property 4.

2.5 A Faster Relaxation by Cutting Plane Generation

An inconvenience of formulation TCNE is that the presence of the f^d and f^p continuous variables may slow down the solution process for large size instances. To obtain an equivalent but faster formulation, we make use of the classical decomposition by Benders (1962). We project out the f^d and f^p variables from the model, as well as the constraints involving those variables. The remaining *master problem* (MP) contains the "complicating" variables x and y, and requires to minimize (2.12) subject to (2.13)–(2.17) and (2.22)–(2.24). Once a feasible solution to MP is found, its values are inserted into the *primal subproblem* (PSP). PSP is a linear problem on the "easy" f^d and f^p variables involving a null objective function and the constraints (2.18)–(2.21) and (2.25).

To solve PSP we make use of the duality theory. We associate variables t_{ij} with constraints (2.18) for all $(i,j) \in A$, v_j with (2.19) for all $j \in V \setminus \{0\}$, w_j with (2.20) for all $j \in P \cup PD$ and with (2.21) for all $j \in D$. We then solve the resulting *dual subproblem* (DSP). The solution of DSP may be either feasible or unbounded. If it is feasible, then it is equal to 0 because of strong duality, so it does not impact on the solution cost of MP (in other words, there are no *Benders optimality cuts* because the costs of the f^d_{ij} and f^p_{ij} variables are 0). If it is unbounded, then let t^r_{ij}, v^r_j, and w^r_j be the values of an extreme ray r of DSP. We multiply the original constraints by the values of the extreme ray and add the corresponding *Benders feasibility cut* to MP. Let R define the set of the extreme rays of DSP. The resulting *Benders-based non-Elementary* (BBNE) formulation is thus:

$$\text{(BBNE)} \quad \min\ z_{BBNE} = \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in P\cup PD} r_j(1-y_j)$$

subject to (2.13)–(2.17), (2.22)–(2.24) and

$$\sum_{(i,j)\in A} (Qt^r_{ij})x_{ij} + \sum_{j\in V\setminus\{0\}} d_j v^r_j + \sum_{j\in P\cup PD} (p_j w^r_j)y_j \le 0 \qquad \forall r \in R. \tag{2.26}$$

Constraints (2.26) are the above mentioned Benders feasibility cuts, and forbid solutions that are infeasible with respect to the removed f^d and f^p variables. The detailed PSP and DSP models are given in Section 2.B.4 of the supplementary material.

Benders feasibility cuts are known to be weak in practice, and our computational tests confirmed this behavior. We could not find improvements of (2.26) based, for example, on special constraint structures (as found for other pickup and delivery problems, see, e.g., Hernández-Pérez and Salazar-González 2004) or on the so-called combinatorial Benders cuts (see, e.g., Codato and Fischetti 2006 and Côté et al. 2014). Thus, we looked for additional valid inequalities that could improve the computational behavior of formulation BBNE.

2.5.1 Valid Inequalities

In this section we present a few families of valid inequalities that strengthen the continuous relaxation of formulation BBNE and improve its computational performance. In the following, for any set $S \subseteq V \setminus \{0\}$, let $d(S) = \sum_{j \in S} d_j$, $p(S) = \sum_{j \in S} p_j$, and $\bar{S} = V \setminus S \setminus \{0\}$. Let also $x(\bar{S} : S) = \sum_{i \in \bar{S}} \sum_{j \in S} x_{ij}$, $x(A(S)) = \sum_{i \in S} \sum_{j \in S} x_{ij}$, and $x(\delta^+(S)) = \sum_{i \in S} \sum_{j \in V \setminus S} x_{ij}$.

First, we impose *subtour elimination constraints* (SECs) as follows:

$$x(A(S)) \leq |S| - 1 \qquad \forall S \subseteq P, \tag{2.27}$$

$$x(\delta^+(S)) \geq 1 \qquad \forall S \subseteq D \cup PD. \tag{2.28}$$

Constraints (2.27) require that at most $|S| - 1$ arcs are selected in a set S containing only pickups, whereas constraints (2.28) impose that at least one arc should be selected among those leaving a set S containing at least a delivery but not the depot. It is worth noting that in Elementary formulations it is easy to transform a SEC taking the "$\leq$" form into one with "$\geq$" form because of degree constraints. However, this is not true for non-Elementary formulations and it is easy to find examples where: (2.28) is not valid for a set $S \subseteq P$ (think of the case in which no pickup is selected in S); and (2.27) is not valid for a set $S \subseteq D \cup PD$ (think of the case in which one or more vertices in S are visited twice).

To obtain SECs involving both sets P and D, we adapt an inequality originally proposed for the symmetric version of the SVRPDSP by Gutiérrez-Jarpa et al. (2009), namely:

$$x(A(S)) \leq |S_D| + \sum_{j \in S_P} y_j - 1 \qquad \forall S = S_D \cup S_P : S_D \subseteq D, S_D \neq \emptyset, S_P \subseteq P, \tag{2.29}$$

where S is partitioned into a subset S_D of delivery customers and a subset S_P of pickup customers. We notice that (2.29) requires S_D to be non-empty, because otherwise the right hand side of the inequality could be negative. We also notice that one could extend the set S by including a subset $S_{PD} \subseteq PD$ of combined customers. However the right hand side of (2.29) would result in $|S_D + S_{PD}| + \sum_{j \in S_P \cup S_{PD}} y_j - 1$, which is very weak and was consequently not used in our tests.

Wolsey (1998) developed a family of constraints to remove subtours in the related context of the TSP-P. The asymmetric version of these constraints directly applies to the SVRPDSP as follows:

$$x(A(S)) \leq \sum_{j \in S \setminus \{k\}} y_j \qquad \forall S \subseteq P, k \in S. \tag{2.30}$$

The number of constraints (2.30) is very large, and even including them in an iterative way, using a branch-and-cut scheme, results in a poor convergence of our models. As done by a number of authors (see, e.g., Laporte and Martello 1990 for the TSP-P, and Gutiérrez-Jarpa et al. 2009 for the symmetric version of the SVRPDSP) we found it convenient to aggregate these constraints, obtaining the following result.

PROPERTY 5 *The following inequality is valid for the SVRPDSP:*

$$x(\delta^+(S)) \geq \sum_{j \in S} \frac{1}{|S|} y_j \qquad \forall S \subseteq P. \tag{2.31}$$

Proof Given in Section 2.A.3 of the supplementary material.

A similar valid inequality, that we could not find in the related literature, is the following one.

PROPERTY 6 *The following inequality is valid for the SVRPDSP:*

$$x(\delta^+(S)) \geq \sum_{j \in S} \frac{p_j}{Q} y_j \qquad \forall S \subseteq P. \tag{2.32}$$

Proof Given in Section 2.A.4 of the supplementary material.

In addition to (2.32), we found other valid inequalities that take into consideration the vehicle capacity. First of all, notice that the vehicle leaves the depot with a residual capacity equal to $Q - d(V)$. Consequently, it cannot visit directly a set S to perform a series of pickups and deliveries that could exceed this residual capacity. This consideration leads to the following result.

PROPERTY 7 *The following inequality (capacity-cut constraint) is valid for the SVRPDSP.*

$$Qx(\bar{S} : S) - \sum_{j \in S} p_j y_j \geq d(V) - d(S) - Q \qquad \forall S \subseteq V \setminus \{0\} : p(S) - d(S) > Q - d(V). \tag{2.33}$$

Proof Given in Section 2.A.5 of the supplementary material.

For the sake of clarity, we notice that there are cases in which a solution may be infeasible for the Benders feasibility cuts (2.26), but feasible for the capacity-cut constraints (2.33).

We finally make use of the two following simple inequalities.

$$\sum_{j \in P \cup PD} p_j y_j \leq Q, \tag{2.34}$$

$$\sum_{i \in V \setminus \{0,j\}} x_{ij} \geq y_j \qquad \forall j \in V \setminus \{0\} : p_j - d_j > Q - d(V). \tag{2.35}$$

Constraint (2.34) is a classical knapsack constraint, whereas constraint (2.35) is an improvement of (2.33) that can be used when S consists of a single vertex j for which $p_j - d_j$ is greater than the residual capacity of the vehicle when leaving the depot. The resulting *two-index non-Elementary* (TINE) formulation is thus:

$$\text{(TINE)} \quad \min \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{j \in P \cup PD} r_j (1 - y_j)$$

$$\text{subject to (2.13)–(2.17), (2.22)–(2.24), and (2.26)–(2.35).}$$

2.5.2 Separation Procedures

Because most of the families of inequalities that we presented in the previous subsection have exponential size, we found it convenient to add them on the fly in a *branch-and-cut* (B&C) fashion. We thus developed tailored separation procedures, that identify those inequalities that are violated by a given solution, so as to add them to the model.

The separation of (2.27) and (2.28) can be done by using a max-flow algorithm in the classical way as follows. Given a (fractional) solution $(\bar{x}, \bar{y})$, we create a *supporting graph* $\bar{G} = \left(\bar{V}, \bar{A}\right)$, where $\bar{V} = V \setminus \{i \in P : \bar{y}_i = 0\}$ and an arc $(i, j) \in \bar{A}$ has capacity equal to $\bar{x}_{ij}$. We then solve a max-flow problem on $\bar{G}$, from 0 to each $i \in P$ having $\bar{y}_i > 0$ to separate (2.27), and from 0 to each $i \in D \cup DP$ for (2.28). The set $\bar{S}$ identified by the min cut and not containing 0 is then checked for detecting a possible violation.

The symmetric version of (2.29) was separated by an enumerative breadth-first-search procedure in Gutiérrez-Jarpa et al. (2009), but we opted for a standard max-flow based procedure. The separation of (2.30) may be performed by adapting to the asymmetric case the procedure described by Wolsey (1998). As discussed in the previous subsection, in our implementation instead of (2.30) we preferred to use the aggregate constraints (2.31) and (2.32). To separate these constraints we adopted a quick check: after having determined during the separation of (2.27) the set $\bar{S}$ leading to the minimum cut, we check whether $\bar{S}$ also violates (2.31) or (2.32). If this is true, then we add the corresponding cut (if both (2.31) and (2.32) are violated, then we only add the most violated cut).

For the new capacity-cut constraints that we introduced, we can state the following result.

PROPERTY 8 *Constraint* (2.33) *can be separated in polynomial time.*

Proof Given in Section 2.A.6 of the supplementary material.

Constraint (2.34) can be improved by using the well known *extended cover inequalities* (ECIs). In our implementation we looked for ECIs in a heuristic way, using the fast algorithm defined in Section 3.3 of Kaparis and Letchford (2010). There are $O(n)$ constraints in (2.35), so we directly add all of them to the model.

2.6 Exact Algorithms

Exact algorithms based on formulations where multiple travels are allowed on arcs have been recently used to solve various routing problems, showing that good results can be obtained in practice. The contributions that we are aware of use these formulations mostly to provide a valid lower bound (see, e.g., Chemla et al. 2013 and Irnich et al. 2015), and just in some cases to attempt to obtain a feasible solution. In this section we extend this idea by showing a few ways in which a non-Elementary formulation can be used iteratively to quickly converge to an optimal solution.

In classical B&C algorithms from the literature the separation procedures are invoked at any node of the enumeration tree, so as to lift as soon as possible the lower bound and fathom the highest amount of nodes. In recent years, however, important advances achieved by commercial MILP solvers made other options computationally effective. These options include the separation of cuts only at integer points using the so-called *lazy callback* (see, e.g., Subramanian et al. 2011 for a related SVPDP), and the old-fashioned cutting plane generation method that solves the MILP to integer optimality before adding cuts (see, e.g., the work on the TSP by Pferschy and Stanek 2014). We consequently attempted several policies for our algorithms, first for solving the SD case with formulation TINE, and then with the aim of using TINE as a basis to optimally solve the CD case.

For what concerns the solution of the SD case with TINE, we found it convenient to use a B&C (defined TINE B&C in the following) that separates inequalities only at integer points, according to the following order: (2.27) (possibly obtaining violated cuts also for (2.31) and (2.32), as mentioned in Section 2.5.2) and (2.28); (2.33); the ECIs associated with (2.34); (2.29) (only for instances having one or more single demand customers); and finally (2.26). The separation of a family of cuts is invoked only if the previous separations failed in providing violated cuts. If more cuts of the same family are found, all of them are added to the model. We also included a simple heuristic algorithm, based on the one proposed by Coelho et al. (2012), to compute a valid upper bound. This algorithm finds a first feasible solution by solving a TSP on the vertices in $V \setminus P$. It then solves a knapsack problem using the pickup demands in $PD \cup P$ as items, and setting the capacity of the knapsack to Q. The pickups selected in the knapsack solution are inserted in greedy way in the TSP tour, one at a time in the lowest-cost position in which they fit, if any.

Note that separation at integer points may be performed by a graph-search algorithm that looks for a path invalidating a given constraint. We implemented a few simple procedures of this kind to separate the above constraints, but we skip their detailed description because they are slight variations of Algorithm 1 in Section 2.B.2. We only mention the fact that, as previously noted in Section 2.4.1, the number of possible paths in a non-Elementary solution increases exponentially with the number of customers visited twice. Thus, if more than six customers are visited twice, we skip the graph-search algorithms and invoke the max-flow based procedures of Section 2.5.2.

For what concerns instead the solution of the CD case, we attempted the following strategies, all based on the TINE B&C that we have just described for the SD case.

***Throw Away B&C* (TA)**. TA includes a simple check in the TINE B&C: for each incumbent solution found during the search process, TA performs a feasibility test to check whether the solution is feasible for the SVRPDSP or requires dropoffs. The test is performed by using procedure CHECKDROPOFF (see Section 2.B.3 of the supplementary material). Infeasible solutions are simply not used by TA, which continues exploring the enumeration tree until an

optimal SVRPDSP solution is found.

***2-Step B&C* (2S)**. During preliminary experiments we found out that the TINE B&C is considerably faster than TA, and for several instances the solution it yields is also optimal for the SVRPDSP. 2S takes advantage of this observation in a two-step procedure. In the first step it solves the TINE B&C, but keeps in memory all the generated cuts along with the best solution that is feasible for the SVRPDSP, if any. If the solution found at the end of this step has no dropoffs, then it is optimal for the SVRPDSP and 2S terminates. Otherwise, 2S proceeds to the second step where it invokes TA. Additionally 2S improves the convergence of TA by initializing it with the best feasible solution found and all the violated cuts generated during the execution of the first step.

***Minimal Extended Network B&C* (MEN)**. MEN works on the basis that solving the extended network (recall Definition 2) always results in a solution with no dropoffs. However, the idea behind MEN is to duplicate as few customers as possible, instead of duplicating all of them as in the classical Elementary formulations. The algorithm starts as in the first step of 2S by solving the TINE B&C and keeping in memory all generated cuts and the best SVRPDSP feasible solution, if any. In case the optimal solution found has no dropoffs, then MEN terminates with an optimal SVRPDSP solution. Otherwise, the solution is inspected to discover which customers have been visited twice. All such customers are duplicated using the procedure in Definition 1. By doing so, we guarantee that the existent dropoffs are removed from the next optimal solution. The next iteration indeed solves the model by considering the updated graph, all the cuts generated at the previous iterations, and the best SVRPDSP feasible solution found so far, if any. The procedure iterates until the optimal solution found has no dropoffs. The procedure checkDropoff used to detect the dropoffs can be time consuming when the number of customers visited twice is large. As speed-up technique, we decided to inspect on the fly only solutions that have less than 10 customer visited twice, and insert the other solutions in a list that is kept sorted by non-increasing cost. At the end of MEN, we select the first entry in the list, if any, and detect if it has dropoffs. We iterate until either a solution with no dropoff is found or the list is empty, thus providing the optimal solution. As mentioned in Section 2.5.1, a solution might be infeasible for the Benders feasibility cuts (2.26) but feasible for the capacity-cut constraints (2.33). Given that MEN is able to catch these infeasibilities by simply duplicating customers, we obtained a small computational improvement by removing the Benders cuts from the version of the TINE B&C used inside MEN.

2.7 Computational Results

We implemented our algorithms on C++ and ran our tests on a PC with an Intel Core i7-3770 3.40 GHz with 8 Gb of RAM. We used Cplex 12.5.1 to solve the MILPs and implement

Table 2.1: Computational results on the single demand case.

			literature				new algorithms					
			SB		GMO*		TCNE		BBNE		TINE	
set	size	#	opt	sec	opt	sec	opt	sec	opt	sec	opt	sec
SB	n=10	24	**24**	0.5	**24**	0.0	**24**	0.1	**24**	0.0	**24**	0.0
	n=20	21	18	516.3	**21**	0.2	**21**	0.5	**21**	0.5	**21**	0.0
	n=30	18	17	621.2	**18**	0.6	**18**	2.3	**18**	3.1	**18**	0.1
	avg/sum	63	59	349.8	**63**	0.2	**63**	0.9	**63**	1.1	**63**	0.0
GMO	25$\leq n \leq$30	18	**18**	13.8	**18**	2.2	**18**	1.5	**18**	4.8	**18**	0.1
	38$\leq n \leq$60	39	26	1691.9	**39**	47.0	**39**	44.1	36	466.4	**39**	3.6
	68$\leq n \leq$90	17	2	3375.9	16	3510.8	**17**	314.6	6	2511.6	**17**	37.3
	avg/sum	74	46	1670.6	73	831.9	**74**	95.9	60	824.0	**74**	10.5

($*$ = run with with Cplex 10 on a Dual Core AMD 2.7 GHz, with 21000 sec time limit)

the B&C algorithms. We selected the default Cplex options, but imposed it to use a single thread to facilitate computational comparison with the literature. Unless stated otherwise, each algorithm was allowed a time limit of one CPU hour for each run. We tested our algorithms on the benchmark instances from the literature and on newly created large size and multiple vehicle instances. Because this amounts to a fairly large number of tests, we provide here only aggregate results, and refer to Section 2.D for detailed results.

2.7.1 Results for the SVRPDSP

Table 2.1 presents the results for the SD case, for which two benchmark sets are available. The first one (called SB set in the following) was proposed by Süral and Bookbinder (2003) and contains 63 instances, among which 24 with 10 customers, 21 with 20 customers, and 18 with 30 customers. The second one (GMO set) was proposed by Gutiérrez-Jarpa et al. (2009) and has 74 instances that we divided into 18 small size, 39 medium size, and 17 large size instances, as shown in Table 2.1. We tested our three formulations, TCNE, BBNE, and TINE (that we recall yielding an optimal solution for the SD case according to Property 1), and compared them with the exact algorithms available in the literature. Namely, the mathematical model by Süral and Bookbinder (2003) (SB), that we re-implemented and ran on our computer with one CPU hour of time limit, and the B&C by Gutiérrez-Jarpa et al. (2009) (GMO), that was run on a Dual Core AMD 2.7 GHz, with 21000 seconds of time limit. Formulation TINE was solved by the TINE B&C of Section 2.6. For each algorithm and group of instances we provide the total number of proven optimal solutions (opt) and the average CPU seconds elapsed (sec). The best opt values are in bold.

Table 2.1 shows that the SB set is very easy. Formulation SB is quite weak and can

solve to proven optimality only 59 instances in this set, whereas all other algorithms solve all instances. The fastest algorithms are GMO and TINE, as they need just fractions of a second. On the GMO set, which is more challenging, algorithm GMO again outperforms SB, but in this case it is not able to solve to proven optimality one instance with 90 customers and requires almost the entire time limit to solve another instance with still 90 customers. On average, formulation TCNE is better than GMO on this set. Formulation BBNE is worse than TCNE, showing that a straight application of the Benders decomposition is not enough to obtain improvements over the two-commodity formulation. The TINE B&C is the fastest algorithm, as it can solve all instances in the GMO set in about 10 seconds on average, and never requiring more than 150 seconds. The only unsolved instance in Gutiérrez-Jarpa et al. (2009) was solved to proven optimality by TINE in just 143 seconds.

Table 2.2 gives the results on the CD case. Here we recall that the solutions of our non-Elementary formulations may have dropoffs, so we use the exact algorithms TA, 2S, and MEN of Section 2.6. Gribkovskaia et al. (2008) introduced the only benchmark set for the CD case (GLS set), that contains 68 instances having between 15 and 100 customers, all requiring a combined demand. We tested the TA, 2S, and MEN algorithms, and compared them with the SB model, the mathematical model by Gribkovskaia et al. (2008) (GLS) that we re-implemented and ran on our PC for one CPU hour, and formulation TCEE of Section 2.3. The breakthrough introduced by the new non-Elementary formulations is evident. Among the Elementary formulations, GLS can solve just one instance to proven optimality, SB 5, and TCEE 24. It is worth noting that formulations SB and GLS (and the one by Gribkovskaia et al. 2007 discussed in Section 2.7.3) were proposed mainly with the purpose of unambiguously describing the problem, so, even with the inclusion of several valid inequalities, their bad performance is not surprising. Notably, the Elementary formulations solve to optimality just one medium size instance and no large size one. This may be imputed to the increase in the size of the underlying extended network. The algorithms based on non-Elementary formulations have a better performance, with TA being able of solving 54 instances, 2S 56, and MEN 67. MEN is the most efficient algorithm, so we ran it with a larger CPU time limit on the only unsolved instance, and could find a proven optimal solution in 17172 seconds (about 4.7 CPU hours).

We also compared the results of our best exact algorithm with the most efficient metaheuristic algorithms from the literature. From the comparison, presented in Section 2.C of the supplementary material, we can clearly see that MEN greatly outperforms the metaheuristics in terms of number of optimal solutions found, time consumption, and optimality gaps. This is another evidence of its efficiency in solving practical problems.

Table 2.2: Computational results on the combined demand case.

		Elementary						non-Elementary					
		GLS		SB		TCEE		TA		2S		MEN	
GLS set	#	opt	sec	opt	sec	opt	sec	opt	sec	opt	sec	opt	sec
$15 \leq n \leq 30$	28	1	3486.5	5	3056.8	23	914.9	**28**	1.7	**28**	0.6	**28**	0.2
$32 \leq n \leq 50$	24	0	t.lim.	0	t.lim.	1	3591.9	18	916.2	18	912.0	**24**	6.5
$71 \leq n \leq 100$	16	0	2903.7	0	t.lim.	0	t.lim.	8	2023.7	10	1386.1	***15**	416.6
avg/sum	68	1	3389.5	5	3376.3	24	2491.5	54	800.2	56	648.3	**67**	100.4

($*$ = remaining instance solved to proven optimality by MEN in 17172 seconds)

2.7.2 Evaluation of the intermediate dropoff relaxation

The main ingredient for the good behavior of the non-Elementary algorithms is the solution of the SVRPDSP-D (i.e., the SVRPDSP relaxation allowing dropoffs). In Table 2.3 we report some more information that allows to get some insight in the quality of this relaxation. The table reports aggregate results for the GLS set, by running the TINE B&C to solve the SVRPDSP-D and MEN to solve the SVRPDSP. The table provides the following information for both algorithms: average number of seconds elapsed (sec); average percentage gaps between the optimal solution value and the lower bounds at the root node before and after adding the cuts of Section 2.5.1 (gap_r and gap_c, respectively); average number of nodes explored by the branching tree (nodes). Moreover, for TINE we provide the number of optimal SVRPDSP-D solutions that were also optimal for the SVRPDSP (feas), and the average percentage gap between the optimal SVRPDSP-D and the SVRPDSP solution values (gap_d). For MEN we also provide the average numbers of customers that were duplicated (dupl) and of iterations that were performed (iter) by the main loop of the algorithm.

Table 2.3: Evaluation of the dropoff relaxation on the combined demand case.

		TINE (for the SVRPDSP-D)							MEN (for the SVRPDSP)						
GLS set	#	opt	sec	gap_r	gap_c	nodes	feas	gap_d	opt	sec	gap_r	gap_c	nodes	dupl	iter
$15 \leq n \leq 30$	28	28	0.3	4.74	0.59	236	26	0.00	28	0.2	4.74	0.59	206	0.1	1.1
$32 \leq n \leq 50$	24	24	10.4	6.27	0.88	6785	24	0.00	24	6.5	6.27	0.98	5209	0.4	1.4
$71 \leq n \leq 100$	16	16	352.6	10.66	0.83	40085	10	0.05	15	416.6	10.70	0.88	32174	1.2	2.2
avg/sum	68	68	86.78	6.67	0.75	11924	60	0.01	67	100.4	6.68	0.79	9493	0.5	1.5

The cuts are very effective, as they manage to reduce by about nine times the root node gap on both problems variants. For the small size instances, 26 out of 28 optimal SVRPDSP-D solutions do not use dropoffs, and the remaining 2 can be made feasible for the SVRPDSP with just two iterations of MEN. For the medium size instances, the SVRPDSP is a bit more

challenging than the SVRPDSP-D and the root node gap after the cuts increases by about 0.1%, but MEN still needs just two iterations in the worst case to close an instance. For the large size instances the difference between the two problems is larger, as confirmed by the value of gap_d that raises to 0.05%. Indeed the SVRPDSP-D is easier as all instances are solved in less than 6 minutes on average by TINE.

One may argue that, in the worst case, MEN would end up duplicating all customers and solving the extended network in the last iteration, which clearly would result in a bad performance of the algorithm. However, in practice we can see that this is not the case. For the large size instances, MEN duplicates about 2 customers on average, and only 4 customers out of 100 in the worst case. These good results confirms a principle already hinted in Subramanian et al. (2011) for a related problem: instead of burdening a formulation in order to enforce conditions that seldom happen in practice, it is better to treat them in a lazier way.

2.7.3 The case of mandatory pickups

We now focus on the special SVRPDSP case arising when all pickups are *mandatory* (see Figure 2.2-(b)). This is known in the literature as the *single vehicle routing problem with pickups and deliveries* (SVRPPD). To solve the SVRPPD we adapted our exact algorithms TA, 2S, and MEN of Section 2.6 as follows: (i) we set $y_j = 1$ for all $j \in P \cup PD$; (ii) we removed the redundant constraints (2.29)–(2.31) and (2.34)–(2.35); (iii) we rounded up to the next integer the right hand side of (2.32), obtaining $x(\delta^+(S)) \geq \left\lceil \sum_{j \in S} p_j/Q \right\rceil$, and the right hand side of (2.33) after having divided the constraint by Q, obtaining $x(\bar{S} : S) \geq \lceil (d(V) + p(S) - d(S))/Q \rceil - 1$. Note that in this case the separation procedure for (2.33) discussed in Proposition 8 is not exact anymore but only heuristic. We call the resulting algorithms TA-MP, 2S-MP, and MEN-MP. We tested these algorithms on the benchmark set proposed by Gribkovskaia et al. (2007), that contains 34 instances with size ranging from 15 to 100 customers (GHLV set in the following).

Table 2.4 compares the results of our exact algorithms with those of the formulation by Gribkovskaia et al. (2007)(GHLV), that we implemented and ran for one CPU hour on our PC. The GHLV formulation performs poorly, solving to proven optimality just one small size instance. Among our algorithms 2S-MP is slightly better than TA-MP, as it solves one more instance to proven optimality. The algorithm having the best performance is MEN-MP, because it can solve to proven optimality all benchmark instances in about 22 seconds on average, and 560 in the worst case. Overall, these results attest the efficiency of our algorithms also for this problem variant, showing that it is possible to extend our approaches to other related 1-M-1 problems obtaining interesting results.

Table 2.4: Computational results on the SVRPPD (i.e., mandatory pickups case).

		literature			new algorithms								
		GHLV			TA-MP			2S-MP			MEN-MP		
GHLV set	#	opt	sec	gap	opt	sec	gap	opt	sec	gap	opt	sec	gap
15≤ n ≤30	14	1	3419.2	10.8	**14**	0.4	0.0	**14**	0.5	0.0	**14**	0.4	0.0
32≤ n ≤50	12	0	t.lim.	16.2	11	559.7	0.1	11	471.3	0.2	**12**	3.6	0.0
71≤ n ≤100	8	0	t.lim.	22.0	7	745.4	0.1	**8**	198.9	0.0	**8**	88.2	0.0
avg/sum	42	1	3525.5	19.9	32	373.1	0.1	33	213.4	0.1	**34**	22.2	0.0

2.7.4 Practical insights on the usage of the SVRPDSP

The SVRPDSP models many practical problems, so it may be interesting for decision makers to understand how important are in practice the main features of this model. To this end we performed two analysis. The first one aims at estimating the savings that can be obtained by allowing mixed deliveries instead of imposing a backhaul distribution. In Table 2.5 we show the average percentage gap between the optimal solution values of the SVRPDSP and of the SVRPDSP with backhauls. By allowing interspersed pickups and deliveries it is possible to save on average 11.05% and 12.7% for instances of sets SB and GMO, respectively. The savings increase when the size of the instances increase, reaching almost 15% for the largest instances of the GMO set. It thus appears very convenient in practice to allow mixed deliveries when possible.

Table 2.5: Average gaps between the optimal solutions of the SVRPDSP and the SVRPDSP with backhauls.

SB set	#	gap	GMO set	#	gap
n=10	24	9.30	25≤ n ≤30	18	12.28
n=20	21	10.27	38≤ n ≤60	39	11.97
n=30	18	14.29	68≤ n ≤90	17	14.80
avg/sum	63	11.05	avg/sum	74	12.70

Our second analysis aims at evaluating the gap in the total driving distance when allowing selective pickups instead of enforcing mandatory pickups, thus directly comparing the SVRPDSP and the SVRPPD. This comparison is affected by the values of the revenues assigned to the pickups, and moreover it is possible only when $Q \geq p(V)$. For the SB set revenues are very high, so all pickups are made even in the selective case, and for the GLS set $Q < p(V)$ for all instances. We thus considered the GMO set, that shows a better balance between costs and revenues and for which 59 out of 74 instances have $Q \geq p(V)$. The results are shown in Table 2.6. On average, more than 8% of the total driving distance can be saved by

allowing selective pickups. This confirms in practice the interest in the selective distribution first mentioned by Golden and Assad (1986).

Table 2.6: Average gap in the total driving distance between the SVRPDSP and the SVRPPD.

GMO set	#	gap
$25 \leq n \leq 30$	15	5.96
$38 \leq n \leq 60$	30	9.74
$68 \leq n \leq 90$	14	7.49
avg/sum	59	8.25

2.7.5 New large size instances

Given that our algorithms were able to provide proven optimal solutions for all benchmark instances in the single vehicle problem variants that we addressed, we decided to better assess the difficulty of the problems by testing our algorithms on new instances. We first attempted instances with asymmetric cost matrices, but did not find relevant differences with respect to the available benchmark instances that have symmetric costs. We then attempted instances with larger numbers of customers, that we created as follows. We selected from the *vehicle routing problem* (VRP) library (`http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html`) 4 large size instances having between 120 to 199 customers each, namely: E121-07c, E135-07f, E151-12b, and E200-16b. For the case of selective pickups, we used the procedure by Gribkovskaia et al. (2008) to convert these VRP instances into SVRPDSP ones, obtaining a new set (BI set) containing 16 instances. For the case of mandatory pickups, we used the procedure in Gribkovskaia et al. (2007) to convert the VRP instances into SVRPPD ones, obtaining an additional set (BI-MP set) with 8 instances. In both sets all customers have combined demands, and this increases their complexity.

We addressed these new sets with our best algorithms for the two problem variants, namely MEN and MEN-MP. The results are given in Table 2.7. For the SVRPDSP, MEN is not able to solve instances with 120 and 135 customers, although the gap remains quite low, around 2% on average. The SVRPPD appears to be easier, as MEN-MP solves all instances with up to 150 customers. The 2 instances with 200 customers are still unsolved, although the gap is around 1%. More research is thus envisaged to find optimal solutions for large size instances.

2.7.6 The case of multiple vehicles

In many real-world scenarios a *fleet* of vehicles instead of a single one is available to serve all customers. We thus decided to extend our study to the case of multiple vehicles, known in the literature as the *multiple vehicle routing problem with deliveries and selective*

Table 2.7: Computational results on large size SVRPDSP and SVRPPD instances.

Selective pickups (SVRPDSP)						Mandatory pickups (SVRPPD)					
		MEN						MEN-MP			
BI set	#	opt	sec	gap	nodes	BI-MP set	#	opt	sec	gap	nodes
n=120	4	1	3208.6	1.6	208570	n=120	2	2	960.3	0.0	42510
n=135	4	0	t.lim.	2.2	91743	n=135	2	2	2257.6	0.0	12980
n=151	4	4	669.6	0.0	35754	n=151	2	2	341.5	0.0	1922
n=200	4	2	2654.0	0.7	49967	n=200	2	0	t.lim.	1.1	46100
avg/sum	16	7	2533.0	1.1	96508	avg/sum	8	6	1789.9	0.3	25878

pickups (MVRPDSP). The MVRPDSP is the generalization of the SVRPDSP in which a fleet of k homogeneous capacitated vehicles is available to perform pickups and deliveries. Transshipment among vehicles is not allowed, and neither are dropoffs and split deliveries or pickups. To our knowledge, the only paper directly approaching the MVRPDSP is the one by Bruck and dos Santos (2012), in which the authors propose a basic three-index mathematical formulation and a hybrid cluster-first heuristic.

To solve the MVRPDSP we adapted our best exact approach, namely the MEN algorithm, as follows. Constraints (2.33) are not valid for the multiple vehicle case, as they are based on the assumption that a vehicle leaves the depot with exactly $d(V)$ units of delivery commodity. We thus replaced them by a set of constraints based on the classical capacity-cut inequalities for the capacitated vehicle routing problem:

$$x(\delta^+(S)) \geq \max\Big(\left\lceil \frac{d(S)}{Q} \right\rceil, \frac{\sum_{i \in S} p_i y_i}{Q} \Big) \qquad \forall S \subseteq V \setminus \{0\} : |S| > 1. \tag{2.36}$$

As for the single vehicle case, inequalities are separated only at integer points. We thus modified the original separation procedures to deal with the new situation and search for subsets possibly leading to violations. For (2.36) this results in a heuristic separation. Moreover, when a set S leading to a violation is found, only the most violated constraint between $x(\delta^+(S)) \geq \lceil d(S)/Q \rceil$ and $x(\delta^+(S)) \geq \sum_{i \in S} p_i y_i / Q$ is added to the model. In case no violation is found, then we separate the Benders cuts (2.26) to ensure feasibility. From now on, let us call MEN-Multi the resulting algorithm.

To analyze the computational behavior of MEN-Multi we modified the SVRPDSP benchmark set GLS by considering a fleet of k vehicles, each having capacity $Q' = \lceil \alpha Q \rceil$, where Q is the vehicle capacity in the original instance. We tested 10 different options for the pair (α, k), namely (0.6, 2), (0.6, 3), (0.6, 4), (0.5, 3), (0.5, 4), (0.5, 5), (0.4, 3), (0.4, 4), (0.4, 5), and (0.4, 6). We obtained in this way a new set of 680 instances, that we call GLS-Multi in the following.

Table 2.8 shows the results of our computational experiments. The algorithm is very efficient and the number of iterations that it requires does not increase significantly with respect to the single vehicle case. For $\alpha = 0.6$ it is able to solve all instances to proven optimality. It also solves 193 instances out of 204 for $\alpha = 0.5$, and 264 instances out of 272 for $\alpha = 0.4$. The average percentage gaps remain very low. This analysis shows how our algorithms and ideas can be successfully extended to problem variants with multiple vehicles. An interesting insight in the problem difficulty is that MEN performs very well for those cases in which capacity constraints are rather loose. This can be noticed by the fact that, for a given value of α, instances get easier for MEN when k increases. For instances with very tight capacities, other approaches such as branch-and-price might prove more efficient. Detailed results are provided in Section 2.D.5 of the supplementary material.

Table 2.8: Computational results of MEN-Multi for the benchmark set GLS-Multi.

		$\alpha = 0.6$				$\alpha = 0.5$				$\alpha = 0.4$			
k	#	opt	sec	gap	iter	opt	sec	gap	iter	opt	sec	gap	iter
2	68	68	111.2	0.00	1.7								
3	68	68	56.2	0.00	1.1	62	454.5	0.07	1.8	63	642.2	0.11	2.1
4	68	68	30.3	0.00	1.1	64	304.4	0.03	1.6	66	494.3	0.04	1.6
5	68					67	315.7	0.00	1.2	67	311.5	0.00	1.3
6	68									68	137.8	0.00	1.3

2.8 Conclusions and Future Research Directions

We studied the class of one-to-many-to-one single vehicle pickup and delivery problems, in which a single capacitated vehicle is used to serve a set of customers requiring a delivery, a pickup, or both. We concentrated on the most studied problem in this class, known as the *single vehicle routing problem with deliveries and selective pickups* (SVRPDSP), in which deliveries are mandatory but pickups are optional and generate a revenue if performed.

Most of the approaches in the literature solve the SVRPDSP by duplicating each customer requiring both a delivery and a pickup into two separate customers, the first requiring only the delivery and the second only the pickup, and then look for Elementary solutions in the resulting extended network. In this work we decided instead to consider the original problem network, and work with solutions that may be non-Elementary. Although this may seem like a simple modification, by allowing non-Elementary solutions the combinatorial structure of the problem changes drastically, and this offers interesting opportunities for the development of efficient exact algorithms.

For the single demand case, where all customers require either a delivery or a pickup, but not both, we developed a formulation looking for Elementary tours on the extended

network and three non-Elementary formulations, two of which were solved by branch-and-cut. Our best algorithm makes use of several families of valid inequalities, and could solve to proven optimality all benchmark instances within a few CPU seconds on a standard PC. For the combined demand case, where customers may require both delivery and pickup, our previous non-Elementary formulations only provide a lower bound to the problem because they allow intermediate dropoffs of commodities. To address this issue we developed three exact algorithms, that extended the previous branch-and-cut by considering different options for removing infeasible solutions. Also in this case our best exact algorithm could solve all benchmark instances, although it required a few CPU hours in the worst case. We then adapted our algorithms to solve the case of mandatory pickups, which turned out to be easier in practice than the SVRPDSP, as our best algorithm could solve all benchmark instances in just a few CPU minutes. We also solved the problem where a fleet of homogeneous vehicles is used, showing that our best exact algorithm is still very efficient, especially when the capacity constraint is loose. These last tests also had the merit to show how our algorithms can be adapted to solve other problem variants.

As these problems are very challenging, it is not difficult to provide instances in which our algorithms fail to provide proven optimal solutions in limited time. Further research is thus envisaged to solve large size instances of the problem variants that we addressed.

The results presented in this chapter indicate that working on the original network of the problem is more challenging than working on the extended network, but worth the effort given the good computational results. Our idea of repeatedly using a non-Elementary formulation to quickly converge to optimality can be applied to a wide range of vehicle routing problems, such as (single or multiple) vehicle routing problems with split deliveries, and vehicle routing problems with load transhipment among vehicles. This research does not have to be limited to one-to-many-to-one pickup and delivery problems, as done in this chapter, but could focus also on many-to-many problems (as the ones addressed in, e.g., Chemla et al. 2013 and Salazar-González and Santos-Hernández 2015), and one-to-one pickup and delivery routing problems with split deliveries (see, e.g., Nowak et al. 2008). This appear to be an interesting and innovative research direction.

Supplementary material 2.A Proofs of Statements

2.A.1 Property 2

PROPERTY 2. *For the CD case formulation TCNE provides a relaxation of the SVRPDSP.*

Proof We prove that TCNE searches a larger solution space than TCEE, by first mapping any feasible TCEE solution into a feasible TCNE solution having the same cost, and then providing a counterexample which is feasible for TCNE but not for TCEE. The statement then follows because TCEE optimally solves the SVRPDSP.

For the first step, let $(\bar{y}_j,\ \bar{x}_{ij},\ \bar{f}^d_{ij},\ \bar{f}^p_{ij})$ be a feasible TCEE solution, and let $(\tilde{y}_j,\ \tilde{x}_{ij},\ \tilde{f}^d_{ij},\ \tilde{f}^p_{ij})$ denote the TCNE solution that we aim to construct. For the y variables, by recalling the notation given in Definition 2, we simply set $\tilde{y}_j = \bar{y}_j$ for $j \in P$ and $\tilde{y}_j = \bar{y}_{\pi(j)}$ for $j \in PD$. For the x variables, we set

$$\tilde{x}_{ij} = \bar{x}_{ij} \qquad \forall\ (i,j) \in A : i,j \in V \setminus PD, \quad (2.37)$$

$$\tilde{x}_{ij} = \bar{x}_{\sigma(i),j} + \bar{x}_{\pi(i),j} \qquad \forall\ (i,j) \in A : i \in PD, j \in V \setminus PD, \quad (2.38)$$

$$\tilde{x}_{ij} = \bar{x}_{i,\sigma(j)} + \bar{x}_{i,\pi(j)} \qquad \forall\ (i,j) \in A : i \in V \setminus PD, j \in PD, \quad (2.39)$$

$$\tilde{x}_{ij} = \bar{x}_{\sigma(i),\sigma(j)} + \bar{x}_{\sigma(i),\pi(j)} + \bar{x}_{\pi(i),\sigma(j)} + \bar{x}_{\pi(i),\pi(j)} \qquad \forall\ (i,j) \in A : i,j \in PD. \quad (2.40)$$

The construction of the $\tilde{f}^d_{ij}$ and $\tilde{f}^p_{ij}$ variables is performed in the same manner. By simple algebraic operations we can check that $\sum_{j \in P \cup PD} r_j(1-\tilde{y}_j) = \sum_{j \in P'} r_j(1-\bar{y}_j)$ and $\sum_{(i,j)\in A} c_{ij}\tilde{x}_{ij} = \sum_{(i,j)\in A'} c_{ij}\bar{x}_{ij}$, so the resulting TCNE solution has the same cost of the initial TCEE one. Coming to the feasibility of constraints (2.13)–(2.25), let us first focus on the vehicle flow conservation imposed by (2.17). When $j \in V \setminus PD$, by applying (2.37)–(2.39), and recalling that $V = \{0\} \cup P \cup D \cup PD$, $D' = D \cup \{\sigma(i) : i \in PD\}$, $P' = P \cup \{\pi(i) : i \in PD\}$, and $V' = \{0\} \cup P' \cup D'$, we get:

$$\begin{aligned}
\sum_{i \in V\setminus\{j\}} (\tilde{x}_{ij} - \tilde{x}_{ji}) &= \sum_{i \in V\setminus PD\setminus\{j\}} (\tilde{x}_{ij} - \tilde{x}_{ji}) + \sum_{i \in PD} (\tilde{x}_{ij} - \tilde{x}_{ji}) \\
&= \sum_{i \in V\setminus PD\setminus\{j\}} (\bar{x}_{ij} - \bar{x}_{ji}) + \sum_{i \in PD} (\bar{x}_{\sigma(i),j} + \bar{x}_{\pi(i),j} - \bar{x}_{j,\sigma(i)} - \bar{x}_{j,\pi(i)}) \\
&= \sum_{i \in D\cup P\cup\{0\}\setminus\{j\}} (\bar{x}_{ij} - \bar{x}_{ji}) + \sum_{i \in D'\setminus D} (\bar{x}_{ij} - \bar{x}_{ji}) + \sum_{i \in P'\setminus P} (\bar{x}_{ij} - \bar{x}_{ji}) \\
&= \sum_{i \in V'\setminus\{j\}} (\bar{x}_{ij} - \bar{x}_{ji})
\end{aligned}$$

which is equal to 0 because $\bar{x}$ satisfies (2.4). When $j \in PD$ we get the same results by applying (2.37), (2.39), and (2.40), so (2.17) is not violated by $\tilde{x}$. Analogous algebraic computations allow to confirm the feasibility of $\tilde{x}$ with respect to the remaining constraints, and this ends the first step of our proof.

For the second step we simply refer to Figure 2.4-(a) of Section 2.4.2, which shows a counterexample which is feasible for TCNE but not for TCEE, because of the above mentioned intermediate dropoff of the load at vertex 1. □

2.A.2 Property 3

PROPERTY 3. *A TCNE solution with split deliveries or pickups can be transformed into a solution having the same cost and for which no delivery or pickup is split.*

Proof Suppose an optimal solution to TCNE is given in which a combined vertex $i \in PD$ is visited twice and its delivery and/or pickup are split. Let d_i^1 and d_i^2, respectively p_i^1 and p_i^2, denote the delivery, respectively pickup, quantities served during the first and second visit to i, respectively. Now build a modified solution in which all the delivery $d_i^1 + d_i^2$ is performed in the first visit, and all the pickup $p_i^1 + p_i^2$ in the second visit. This modification leaves unchanged the load on the vehicle from 0 to the first visit to i and from the second visit to i to 0, and decreases it by $d_i^2 + p_i^1$ from the first to the second visit to i. The new solution is still feasible, has no split in i, and has cost equal to the original one. The process can be repeated until all deliveries and pickups are not split. □

2.A.3 Property 5

PROPERTY 5. *The following inequality is valid for the SVRPDSP:*

$$x(\delta^+(S)) \geq \sum_{j \in S} \frac{1}{|S|} y_j \qquad \forall S \subseteq P. \tag{2.31}$$

Proof We could use an algebraic proof similar to that of Gutiérrez-Jarpa et al. (2009) for the symmetric case, but we opted to proceed in a simpler fashion. First notice that there are two mutually exclusive cases for the vehicle flow that leaves a subset S of pickup vertices, that can be represented by the two following inequalities:

$$x(\delta^+(S)) = 0 \qquad \forall S \subseteq P : \sum\nolimits_{j \in S} y_j = 0, \tag{2.41}$$

$$x(\delta^+(S)) \geq 1 \qquad \forall S \subseteq P : \sum\nolimits_{j \in S} y_j \geq 1. \tag{2.42}$$

Notice also that $\sum_{j \in S} \frac{1}{|S|} y_j = 0$ when $\sum_{j \in S} y_j = 0$, and $0 < \sum_{j \in S} \frac{1}{|S|} y_j \leq 1$ when $\sum_{j \in S} y_j \geq 1$, so (2.31) is valid on both the two mutually exclusive cases and the statement follows. □

2.A.4 Property 6

PROPERTY 6. *The following inequality is valid for the SVRPDSP:*

$$x(\delta^+(S)) \geq \sum_{j \in S} \frac{p_j}{Q} y_j \qquad \forall S \subseteq P. \tag{2.32}$$

Proof We proceed again by making use of the two mutually exclusive cases considered in the previous proof of Property 5. The validity of (2.32) comes from the fact that: (i) $\sum_{j \in S} \frac{p_j}{Q} y_j = 0$ when $\sum_{j \in S} y_j = 0$, so (2.41) is satisfied; (ii) $\sum_{j \in S} \frac{p_j}{Q} y_j > 0$ and, because of (2.34), $\sum_{j \in S} \frac{p_j}{Q} y_j \leq 1$, so also (2.42) is satisfied. Constraint (2.32) is thus valid for any possible set $S \subseteq P$. □

2.A.5 Property 7

PROPERTY 7. *The following inequality (capacity-cut constraint) is valid for the SVRPDSP.*

$$Qx(\bar{S} : S) - \sum_{j \in S} p_j y_j \geq d(V) - d(S) - Q \qquad \forall S \subseteq V \setminus \{0\} : p(S) - d(S) > Q - d(V). \tag{2.33}$$

Proof First recall that according to our notation $d(S) = \sum_{j \in S} d_j$, $p(S) = \sum_{j \in S} p_j$, and $x(\bar{S} : S) = \sum_{i \in \bar{S}} \sum_{j \in S} x_{ij}$. We can rewrite (2.33) as

$$x(\bar{S} : S) \geq \frac{d(V) - d(S)}{Q} + \frac{\sum_{j \in S} p_j y_j}{Q} - 1 \qquad \forall S \subseteq V \setminus \{0\} : p(S) - d(S) > Q - d(V). \tag{2.43}$$

Note that $\frac{d(V)-d(S)}{Q} \leq 1$ because $d(V) - d(S) \leq d(V) \leq Q$, and $\frac{\sum_{j \in S} p_j y_j}{Q} \leq 1$ because of (2.34), so the right hand side of (2.43) takes a value that is at most one. Consequently, (2.43) imposes that the vehicle flow that goes from $\bar{S}$ to S is at least one for those sets S and solutions y satisfying $d(V) - d(S) + \sum_{j \in S} p_j y_j > Q$ (and it is instead loose if $d(V) - d(S) + \sum_{j \in S} p_j y_j \leq Q$).

To see that this condition is always satisfied, we consider two mutually exclusive cases:

1. the vehicle leaves the depot and first visits $\bar{S}$. In this case, for connectivity, the vehicle is forced at some point to travel from $\bar{S}$ to S, so the vehicle flow from $\bar{S}$ to S is at least one and (2.43) is satisfied;

2. the vehicle leaves the depot and first visits S. In this case notice that the vehicle leaves the depot with a load equal to $d(V)$. Thus, if it visited all the vertices in S performing the pickups described by y, its load when finally leaving S would be $d(V) - d(S) + \sum_{j \in S} p_j y_j$, but that would exceed the vehicle capacity Q because of the above assumption on S and y. Thus, the vehicle is forced to leave S at some point to perform some deliveries in

$\bar{S}$ and increase its residual loading space, and then return to S later on. Consequently, also in this case (2.43) is satisfied and that proves the statement. □

2.A.6 Property 8

PROPERTY 8. *Constraint* (2.33) *can be separated in polynomial time.*

Proof We are given a possibly fractional solution $(\bar{x}, \bar{y})$ to formulation TINE. Recall that $p(S) = \sum_{j \in S} p_j$, and let us use the following additional notation: $y(S) = \sum_{j \in S} p_j \bar{y}_j$ and $\underline{y}(S) = \sum_{j \in S} p_j (1 - \bar{y}_j)$. We first partition $p(S)$ as

$$p(S) = \sum_{j \in S} p_j \bar{y}_j + \sum_{j \in S} p_j (1 - \bar{y}_j) = y(S) + \underline{y}(S). \tag{2.44}$$

We also partition the whole pickups as $p(V) = p(S) + p(\bar{S})$. By including this in (2.44) we get

$$y(S) = p(S) - \underline{y}(S) = p(V) - p(\bar{S}) - \underline{y}(S). \tag{2.45}$$

Let us now rewrite constraints (2.33) as

$$x(S : \bar{S}) \geq \frac{d(V)}{Q} - \frac{d(S)}{Q} + \frac{y(S)}{Q} - 1. \tag{2.46}$$

Then, by using (2.45), we can rewrite (2.46) as

$$x(S : \bar{S}) \geq \frac{d(V)}{Q} - \frac{d(S)}{Q} + \frac{p(V) - p(\bar{S}) - \underline{y}(S)}{Q} - 1,$$

so we obtain

$$x(S : \bar{S}) + \frac{d(S)}{Q} + \frac{p(\bar{S})}{Q} + \frac{\underline{y}(S)}{Q} \geq \frac{d(V)}{Q} + \frac{p(V)}{Q} - 1. \tag{2.47}$$

Consequently, (2.47) is equivalent to (2.33). We now present an algorithm to exactly separate (2.47). We start by the supporting graph already used for the separation of (2.27) and (2.28), namely, $\bar{G} = \big(\bar{V}, \bar{A}\big)$, where $\bar{V} = V \setminus \{i \in P : \bar{y}_i = 0\}$ and an arc $(i, j) \in \bar{A}$ has capacity equal to $\bar{x}_{ij}$. Once again we solve a series of max-flow problems, one for each vertex $i \in \bar{V}$, but in this case we first need to modify $\bar{G}$ to obtain a new supporting graph for each i. This new graph, that we define $\bar{G}(i) = \big(\bar{V}(i), \bar{A}(i)\big)$, is built as follows:

- copy $\bar{G}$ in $\bar{G}(i)$;
- remove vertex 0 from $\bar{G}(i)$;
- add a new vertex $n + 1$ to $\bar{G}(i)$;
- connect $n+1$ to all vertices $j \in \bar{V}(i) \setminus \{n+1\}$ with arcs of capacity equal to $\frac{d_j}{Q} + \frac{p_j(1-\bar{y}_j)}{Q}$;

- connect all vertices $j \in \bar{V}(i) \setminus \{i\}$ to i with arcs of capacity equal to $\bar{x}_{ji} + \frac{p_j}{Q}$.

Now compute the maximum flow from $n+1$ to i in $\bar{G}(i)$, and let f be the resulting flow value. Let also S be the subset identified by the min cut and containing vertex i, and $\bar{S}$ the remaining subset containing $n+1$. If $f < \frac{d(V)}{Q} + \frac{p(V)}{Q} - 1$, then the inequality (2.47) is violated by the current set S, and can be added to the model. Otherwise, we proceed with the next vertex i, until either a violation is detected or max-flow computations have been performed for all vertices.

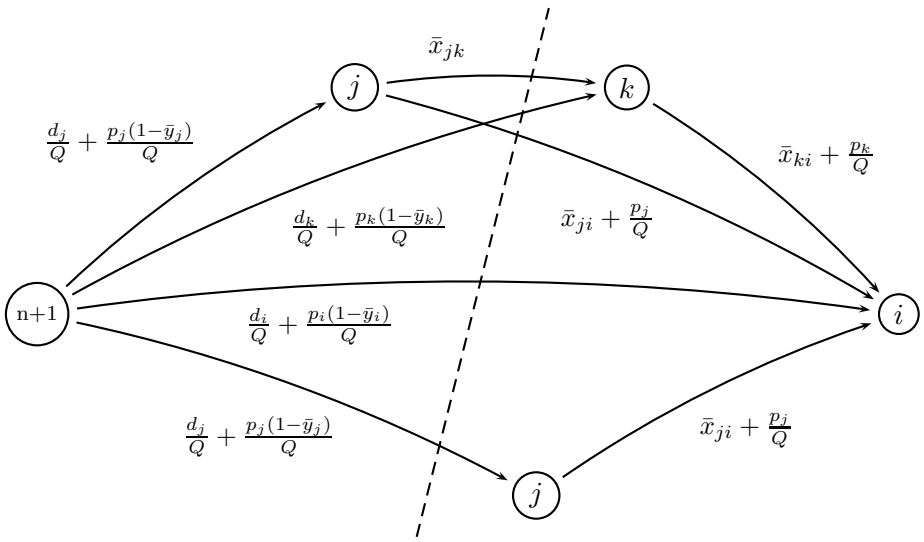


Figure 2.5: *Max-flow separation procedure for capacity-cut constraints* (2.33).

To see that this procedure exactly separates (2.47), let us focus on the example depicted in Figure 2.5. The value f of the maximum flow is equal to that of the min cut, represented by the dashed line in the figure. Set S is at the right of the min cut, and $\bar{S}$ at the left. For each customer $j \in \bar{V}(i) \setminus \{i, n+1\}$, there are two possible cases:

1. if $j \in S$, then by construction its contribution to f is given by $\frac{d_j + p_j(1-\bar{y}_j)}{Q}$;
2. if instead $j \in \bar{S}$, then its contribution to f is equal to $\frac{p_j}{Q} + \sum_{k \in S} \bar{x}_{jk}$.

Hence, summing up the contributions from all vertices, we obtain

$$f = \sum_{j \in \bar{S}} \frac{p_j}{Q} + \sum_{j \in \bar{S}} \sum_{k \in S} \bar{x}_{jk} + \sum_{j \in S} \left(\frac{d_j + p_j(1-\bar{y}_j)}{Q} \right) = x(\bar{S} : S) + \frac{D(S)}{Q} + \frac{P(\bar{S})}{Q} + \frac{y(S)}{Q}.$$

The correctness of the procedure follows from the fact that f is equivalent to the left hand side of (2.47), so if $f < \frac{d(V)}{Q} + \frac{p(V)}{Q} - 1$, then S induces a violated cut, otherwise no violation exists. For the sake of clarity notice that, in case a violation is found, we are sure that the condition imposed in (2.33), i.e., $p(S) - d(S) > Q - d(V)$, is satisfied. This can be observed by looking at the rewritten form of the capacity-cut constraints (2.46). In case

$p(S) - d(S) \leq Q - d(V)$, the right hand side of (2.46) is non-positive and thus the inequality cannot be violated. □

Supplementary material 2.B Additional details on the formulations and on the implemented algorithms

2.B.1 Example of a TCNE solution with an arc being traversed twice

Due to the fact that TCNE works on the original network and multiple visits are allowed to CD customers, it might happen that an arc between two CD customers, say i and j, is traversed twice, i.e., $x_{ij} = 2$. In this case the associated total flow passing through the arc is the sum of the flows in the first and second traversals.

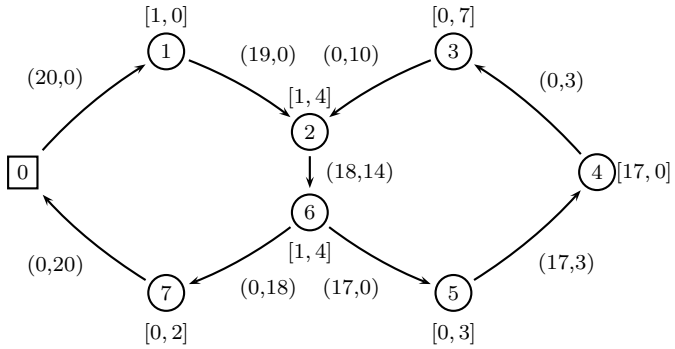
To better illustrate this situation, consider the example depicted in Figure 2.6. Vertex 0 represents the depot and vertices 1 to 7 are the customers. The notation is the same one adopted in the paper. The vehicle capacity is $Q = 20$. In Figure 2.6-(a) arc $(2, 6)$ is being traversed twice with a total flow $f^d_{26} + f^p_{26} = 32$, which is feasible because it does not exceed $2Q$ (refer to constraints (2.18)). In Figure 2.6-(b) the flow is disaggregated in the two traversals, (18,0) during the first traversal and (0,14) during the second one. Notice that, as expected, in neither of them the vehicle capacity is violated. The flow values can be obtained by using procedure REMOVESPLIT, described in the next section.

2.B.2 Procedure for removing split deliveries or pickups

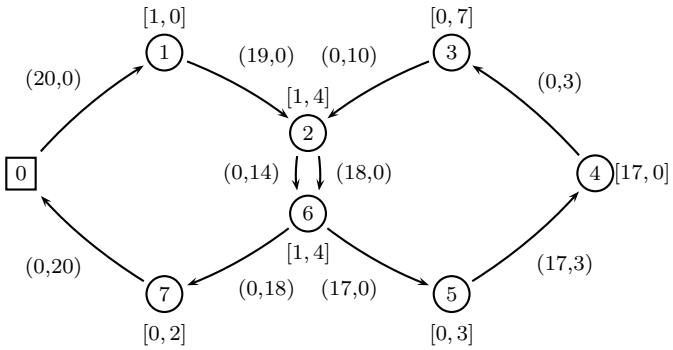
We are given a solution to formulation TCNE that possibly contains split deliveries or pickups. Following Property 3, we want to compute the flows of delivery and pickup commodity, f^d_{ij} and f^p_{ij}, that do not require any split delivery or pickup. This can be obtained by making use of the recursive procedure REMOVESPLIT given in Algorithm 1.

In details, let $\tilde{x}_{ij}$ and $\tilde{y}_j$ denote the values taken by the corresponding variables in the TCNE solution, and compute the number of times in which a vertex j has been visited as $\tilde{\delta}_j = \sum_{i \in V} \tilde{x}_{ij}$. Initialize an array *visits*[j] to 0 for all vertices $j \in N$, and set to 0 also the value of the current vertex (*current*, in the algorithm), of the pickup load (*pLoad*), and of the delivery load (*dLoad*). Then invoke REMOVESPLIT. This procedure attempts to build a path that starts and ends at the depot, and visits each customer j exactly $\tilde{\delta}_j$ times. At each forward step it connects the path to a vertex i that can be reached from *current* (there can be up to two such vertices), and updates the flows by:

- performing the full delivery of i if i is visited for the first time;



(a) Aggregated flows on arc $(2, 6)$



(b) Disaggregated flows on arc $(2, 6)$

Figure 2.6: *Example of a TCNE solution with an arc being traversed twice.*

- performing the full pickup of i if $\tilde{y}_i = 1$ and i is visited just once, or if i is visited for the second time (in this second case $\tilde{y}_i$ is surely 1).

In the backtrack step the flows are set back to 0. The array *visits* is updated accordingly. When the path returns at the depot, a quick check is performed by means of a simple procedure called ALLVISITED, not reported in explicit, which returns *true* if $visits[i]{=}\tilde{\delta}_i$ for all $i \in N$, and *false* otherwise. The first time that ALLVISITED returns *true*, the value of the flag *isDone* is set to *true* and consequently REMOVESPLIT terminates. Note that the complexity of REMOVESPLIT grows exponentially with the number of vertices visited twice, because each such vertex may lead to two different recursive calls. This has not been an issue in our tests, but, in case of very large instances, one could alternatively remove splits by means of the procedure outlined in the next section, based on a MILP formulation.

2.B.3 Procedure for detecting intermediate dropoffs

The exact algorithms that we implemented for the CD case are based on solutions of the TINE formulation, and eventually need to detect if these solutions require dropoffs. This

Algorithm 1 remove split deliveries or pickups from a TCNE solution.

```
1: function removeSplit(current, dLoad, pLoad, visits)
2:     if current = 0 and visits[0] ≠ 0 then
3:         if allVisited(visits) then return true
4:         else return false
5:     end if
6:     for each i ∈ N : x̃_{current,i} > 0 do
7:         if visits[i]=0 then
8:             dLoad ← dLoad − d_i
9:             if (ỹ_i = 1 and δ̃_i = 1) then pLoad ← pLoad + p_i
10:        else
11:            pLoad ← pLoad + p_i
12:        end if
13:        f^d_{current,i} ← dLoad;
14:        f^p_{current,i} ← pLoad
15:        visits[i] ← visits[i] + 1
16:        isDone ← removeSplit(i, dLoad, pLoad, visits)                ▷ forward
17:        if isDone then
18:            return true
19:        else                                                         ▷ backtrack
20:            f^d_{current,i} ← 0;
21:            f^p_{current,i} ← 0
22:            visits[i] ← visits[i] − 1
23:        end if
24:    end for
25:    return false
26: end function
```

can be achieved by an easy adaptation of Algorithm 1 presented in the previous section, or through a MILP based procedure. The adaptation of Algorithm 1 still has a complexity that grows exponentially with the number of customers visited twice. Here we focus on the description of the MILP based procedure, called checkDropoff, that is more convenient for large size instances.

In particular, let $(\bar{x}, \bar{y})$ be a solution of the TINE formulation (i.e., minimize (2.12) subject to (2.13)–(2.17), (2.22)–(2.24), and (2.26)–(2.35)), and recall that dropoffs may only happen at customers visited twice. The idea of checkDropoff is to determine which customers are visited twice in $(\bar{x}, \bar{y})$, create a new graph in which these customers (and only these customers) are duplicated, and then solve on this graph a version of the TCEE formulation (i.e., minimize (2.1) subject to (2.2)-(2.11)) that embeds the information from $(\bar{x}, \bar{y})$. The outcome is a feasible solution, if any, with f^p and f^d commodity flows requiring no dropoffs,

or a proof of infeasibility of $(\bar{x}, \bar{y})$.

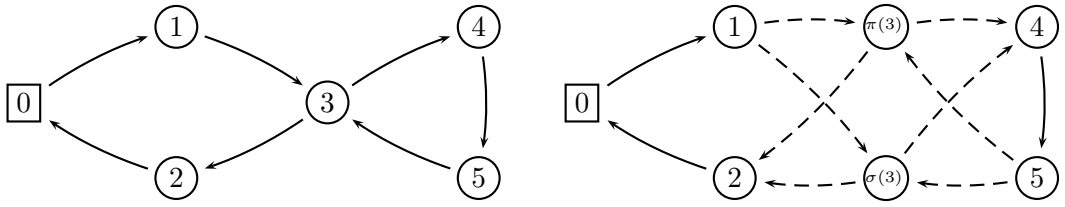
Let $\overline{PD}_2$ be the set of customers visited twice in $(\bar{x}, \bar{y})$ and $\overline{PD}_1 = PD \setminus \overline{PD}_2$ the set of those visited once. We create a new graph $\widetilde{G} = \{\widetilde{V}, \widetilde{A}\}$, by applying the duplication in Definition 1 to all $i \in \overline{PD}_2$, and by setting all customers $i \in \overline{PD}_1$ as pure delivery customers with demand $d_i - p_i\bar{y}_i$. We then embed the information from $(\bar{x}, \bar{y})$ by setting $y_i = \bar{y}_i$ for all $i \in P$, and imposing on each arc (i, j) the following constraint:

- $x_{ij} = \bar{x}_{ij}$ if $i, j \in \widetilde{V} \setminus \overline{PD}_2$;
- $x_{\pi(i),j} + x_{\sigma(i),j} = \bar{x}_{ij}$ if $i \in \overline{PD}_2, j \in \widetilde{V} \setminus \overline{PD}_2$;
- $x_{i,\pi(j)} + x_{i,\sigma(j)} = \bar{x}_{ij}$ if $i \in \widetilde{V} \setminus \overline{PD}_2, j \in \overline{PD}_2$;
- $x_{\pi(i),\pi(j)} + x_{\pi(i),\sigma(j)} + x_{\sigma(i),\pi(j)} + x_{\sigma(i),\sigma(j)} = \bar{x}_{ij}$ if $i, j \in \overline{PD}_2$.

By doing this we impose TCEE to follow the tour induced by $\bar{x}_{ij}$, but let it find the best way to visit and serve the customers in $\overline{PD}_2$. If TCEE finds a feasible solution on $\widetilde{G}$, then checkDropoff easily maps the resulting f^p and f^d flows into the original graph and returns the feasible solution. Otherwise, it returns the proof of infeasibility.

A simple example of the way $\widetilde{G}$ is created is given in Figure 2.7. Figure 2.7-(a) represents the original $(\bar{x}, \bar{y})$ solution, where any arc (i, j) in solid lines indicate that $\bar{x}_{ij} = 1$. We then duplicate customer 3 creating $\pi(3)$ and $\sigma(3)$. Figure 2.7-(b) depicts the resulting graph $\widetilde{G}$, again omitting arcs that are not used in the solution. Notice that dashed lines represent new arcs, whose associated variable values will be fixed by TCEE, whereas the values of x_{01}, x_{45}, and x_{20} are set to 1.

It is interesting to notice that this procedure can be applied to find the values of flow variables f^d and f^p that do not require split deliveries or pickups, so it is also an alternative to Algorithm 1 outlined in the previous section.



(a) Original non-Elementary solution (b) Supporting graph $\widetilde{G}$

Figure 2.7: Creation of the supporting graph used to detect dropoffs.

2.B.4 Details of the Benders Decomposition

In this section we give the details of the primal subproblem (PSP) and the dual subproblem (PSP) used in the Benders decomposition of Section 2.5. Recall that $V = \{0\} \cup P \cup D \cup PD$. Given a solution $(\bar{x}, \bar{y})$ to the master problem (minimize (2.12) subject to (2.13)–(2.17) and (2.22)–(2.24)), the PSP can be modeled by

$$\text{(PSP)} \quad \min\ 0 \tag{2.48}$$

subject to:

$$f^d_{ij} + f^p_{ij} \le Q\bar{x}_{ij} \qquad \forall\ (i,j) \in A, \tag{2.49}$$

$$\sum_{i \in V \setminus \{j\}} \left(f^d_{ij} - f^d_{ji}\right) = d_j \qquad \forall\ j \in V \setminus \{0\}, \tag{2.50}$$

$$\sum_{i \in V \setminus \{j\}} \left(f^p_{ji} - f^p_{ij}\right) = p_j \bar{y}_j \qquad \forall\ j \in P \cup PD, \tag{2.51}$$

$$\sum_{i \in V \setminus \{j\}} \left(f^p_{ji} - f^p_{ij}\right) = 0 \qquad \forall\ j \in D, \tag{2.52}$$

$$f^d_{ij}, f^p_{ij} \ge 0 \qquad \forall\ (i,j) \in A. \tag{2.53}$$

The corresponding DSP can be modeled by associating variables t_{ij} with constraints (2.49) for $(i,j) \in A$, v_j with (2.50) for $j \in V \setminus \{0\}$, w_j with (2.51) for $j \in P \cup PD$ and with (2.52) for $j \in D$, so obtaining:

$$\text{(DSP)} \quad \max \sum_{(i,j) \in A} (Q\bar{x}_{ij}) t_{ij} + \sum_{j \in V \setminus \{0\}} d_j v_j + \sum_{j \in P \cup PD} (p_j \bar{y}_j) w_j \tag{2.54}$$

subject to:

$$t_{ij} - v_i + v_j \le 0 \qquad \forall\ i,j \in V \setminus \{0\}, i \ne j, \tag{2.55}$$

$$t_{ij} - w_j + w_i \le 0 \qquad \forall\ i,j \in V \setminus \{0\}, i \ne j, \tag{2.56}$$

$$t_{0i} + v_i \le 0 \qquad \forall\ i \in V \setminus \{0\}, \tag{2.57}$$

$$t_{i0} - v_i \le 0 \qquad \forall\ i \in V \setminus \{0\}, \tag{2.58}$$

$$t_{0i} - w_i \le 0 \qquad \forall\ i \in V \setminus \{0\}, \tag{2.59}$$

$$t_{i0} + w_i \le 0 \qquad \forall\ i \in V \setminus \{0\}, \tag{2.60}$$

$$t_{ij} \le 0 \qquad \forall\ (i,j) \in A, \tag{2.61}$$

$$v_i, w_i \gtrless 0 \qquad \forall\ i \in V \setminus \{0\}. \tag{2.62}$$

Supplementary material 2.C Computational Comparison with existing metaheuristic algorithms

In this section we compare the results of our best exact algorithm (MEN) with some efficient metaheuristic algorithms. We selected for the comparison the most effective algorithms in the literature, that are, according to our knowledge, the *general variable neighborhood search* (GVNS) by Coelho et al. (2012), and the *evolutionary algorithm* (EA) by Bruck et al. (2012). Both algorithms were run only on the GLS set and executed several times with different random seeds. GVNS was run 30 times, each with a limited number of iterations on an Intel i7 2.93GHz with 8Gb of RAM, and EA 10 times, each with a time limit of 3 hours on an Intel i7 3.07GHz with 6Gb of RAM.

The results of our comparison are given in Table 2.9. Columns gap_b and gap_{avg} give, respectively, the average percentage gaps between the best and average solution values found by the metaheuristic and the optimal solution value. Similarly, column *gap* gives the average percentage gap between the best upper bound found by MEM and the optimal solution value.

From the results we can clearly see that MEN greatly outperforms the heuristics in terms of number of optimal solutions found, time consumption, and optimality gaps. This is another strong evidence of its efficiency in practice.

Table 2.9: Comparison between MEN and the best metaheuristics from the SVRPDSP literature.

		EA				GVNS				MEN		
GLS set	#	opt	sec	gap_b	gap_{avg}	opt	sec	gap_b	gap_{avg}	opt	sec	gap
$15 \leq n \leq 30$	28	14	518.6	0.3	1.3	20	22.3	1.0	2.9	28	0.2	0.0
$32 \leq n \leq 50$	24	1	5138.2	0.8	1.2	10	109.0	0.5	1.7	24	6.5	0.0
$71 \leq n \leq 100$	16	0	22117.3	11.2	12.7	5	1156.4	1.4	3.3	15	416.6	0.0
avg/sum	68	15	7231.1	3.0	3.9	35	319.7	0.9	2.5	67	100.4	0.0

Supplementary material 2.D Detailed Computational Results

Due to the large number of tests, in Section 2.7 of the paper we presented only aggregate results. Here we give instead detailed results for each run of our algorithms on each instance. All tests had a time limit of 1 CPU hour on a PC equipped with an Intel Core i7-3770 3.40 GHz with 8 Gb of RAM, with the exception of the GMO branch-and-cut, that was run on a Dual Core AMD 2.7 GHz, with 21000 seconds of time limit. The columns in the tables have the following meanings:

- Column *instance* gives the name of the instance;

- Column z_{opt} reports the optimal solution value, if known;
- For a given algorithm A, producing a lower bound L_A and an upper bound U_A, column *gap* gives the percentage gap of A computed as $100(U_A - L_A)/U_A$. The symbol '$-$' means that the gap is 0 and the algorithm has found a proven optimal solution;
- Column *sec* reports the number of seconds required by the algorithm to run to completion, 't.lim.' indicates that the algorithm reached the time limit, and 'm.lim.' that it stopped because of memory limit; (for the results on the instances that refer to Table 2.1, that are very easy, *sec* has two digits, whereas for all other tests it has one digit as in the paper);
- The additional columns in Tables 2.19, 2.20, and 2.21 have the same meaning as those reported in Table 2.3 in the paper, but refer to single instances instead of aggregate results. Namely: columns gap_r and gap_c give, respectively, the percentage gaps between the optimal solution value and the lower bounds at the root node before and after adding the cuts of Section 2.5.1; column feas reports if the solution found for the SVRPDSP-D is also feasible for the SVRPDSP (feas=1) or not (feas=0); column gap_d gives the percentage gap between the optimal SVRPDSP-D and the SVRPDSP solution values; columns iter and dupl give, respectively, the number of customers that were duplicated and of iterations that were performed by MEN;
- For the large size instances in Tables 2.25 and 2.26, column UB_{best} reports the best known upper bound (which is equal to z_{opt} when the gap is 0).

2.D.1 Detailed results for the SVRPDSP

Table 2.10: Detailed results for Table 2.1: SB set, n=10 ($n = |P| + |D|$).

				SB		GMO		TCNE		BBNE		TINE	
instance	$\|P\|$	$\|D\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
E8_1_10_20	2	8	166	—	0.01	—	0.02	—	0.05	—	0.00	—	0.00
E8_1_10_30	3	7	166	—	0.02	—	0.02	—	0.04	—	0.00	—	0.00
E8_1_10_40	4	6	166	—	0.01	—	0.02	—	0.06	—	0.01	—	0.00
E8_31_40_20	2	8	233	—	0.03	—	0.03	—	0.08	—	0.00	—	0.00
E8_31_40_30	3	7	233	—	0.02	—	0.03	—	0.09	—	0.01	—	0.00
E8_31_40_40	4	6	233	—	0.03	—	0.03	—	0.09	—	0.01	—	0.00
E10_1_10_20	2	8	173	—	0.00	—	0.02	—	0.04	—	0.00	—	0.00
E10_1_10_30	3	7	173	—	0.00	—	0.02	—	0.06	—	0.00	—	0.00
E10_1_10_40	4	6	173	—	0.01	—	0.02	—	0.03	—	0.01	—	0.00
E10_81_90_20	2	8	167	—	0.00	—	0.02	—	0.03	—	0.00	—	0.00
E10_81_90_30	3	7	167	—	0.00	—	0.02	—	0.07	—	0.01	—	0.00
E10_81_90_40	4	6	167	—	0.00	—	0.03	—	0.03	—	0.00	—	0.00
F10_1_10_20	2	8	208	—	0.08	—	0.04	—	0.09	—	0.02	—	0.00
F10_1_10_30	3	7	258	—	0.08	—	0.11	—	0.15	—	0.02	—	0.01
F10_1_10_40	4	6	296	—	0.07	—	0.22	—	0.13	—	0.03	—	0.00
F12_1_10_20	2	8	92	—	3.77	—	0.03	—	0.05	—	0.01	—	0.00
F12_1_10_30	3	7	92	—	3.64	—	0.03	—	0.07	—	0.01	—	0.00
F12_1_10_40	4	6	92	—	3.20	—	0.03	—	0.08	—	0.01	—	0.00
F12_81_90_20	2	8	242	—	0.10	—	0.04	—	0.07	—	0.01	—	0.00
F12_81_90_30	3	7	242	—	0.08	—	0.04	—	0.09	—	0.01	—	0.00
F12_81_90_40	4	6	242	—	0.10	—	0.03	—	0.11	—	0.01	—	0.00
M1_10_20	2	8	3154	—	0.03	—	0.04	—	0.09	—	0.01	—	0.00
M1_10_30	3	7	3154	—	0.03	—	0.03	—	0.12	—	0.01	—	0.00
M1_10_40	4	6	3154	—	0.04	—	0.03	—	0.09	—	0.01	—	0.00

Table 2.11: Detailed results for Table 2.1: SB set, n=20 ($n = |P| + |D|$).

				SB		GMO		TCNE		BBNE		TINE	
instance	$\|P\|$	$\|D\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
E8_11_30_20	4	16	254	—	0.01	—	0.03	—	0.21	—	0.01	—	0.01
E8_11_30_30	6	14	254	—	0.01	—	0.03	—	0.26	—	0.03	—	0.01
E8_11_30_40	8	12	254	—	0.01	—	0.03	—	0.17	—	0.01	—	0.01
E10_11_30_20	4	16	272	—	0.12	—	0.04	—	0.22	—	0.04	—	0.02
E10_11_30_30	6	14	272	—	0.10	—	0.04	—	0.38	—	0.02	—	0.01
E10_11_30_40	8	12	272	—	0.14	—	0.05	—	0.27	—	0.04	—	0.01
E10_41_60_20	4	16	267	—	0.24	—	0.48	—	0.55	—	0.11	—	0.02
E10_41_60_30	6	14	267	—	0.26	—	0.47	—	0.43	—	0.21	—	0.01
E10_41_60_40	8	12	267	—	0.22	—	0.33	—	0.43	—	0.13	—	0.03
F10_11_30_20	4	16	531	—	6.17	—	0.14	—	0.77	—	1.53	—	0.03
F10_11_30_30	6	14	531	—	10.00	—	0.14	—	0.78	—	0.83	—	0.03
F10_11_30_40	8	12	582	—	21.61	—	0.59	—	2.00	—	2.31	—	0.08
F12_11_30_20	4	16	156	—	1.23	—	0.22	—	0.63	—	0.46	—	0.04
F12_11_30_30	6	14	156	—	0.60	—	0.19	—	0.82	—	0.86	—	0.03
F12_11_30_40	8	12	156	—	0.78	—	0.21	—	0.54	—	1.01	—	0.03
F12_41_60_20	4	16	86	13.79	t.lim.	—	0.09	—	0.32	—	0.25	—	0.03
F12_41_60_30	6	14	86	13.92	t.lim.	—	0.09	—	0.47	—	0.81	—	0.02
F12_41_60_40	8	12	86	12.80	t.lim.	—	0.17	—	0.33	—	0.65	—	0.03
M5_24_20	4	16	4263	—	0.28	—	0.10	—	0.37	—	0.09	—	0.01
M5_24_30	6	14	4263	—	0.22	—	0.10	—	0.33	—	0.21	—	0.02
M5_24_40	8	12	4263	—	0.15	—	0.10	—	0.30	—	0.11	—	0.01

Table 2.12: Detailed results for Table 2.1: SB set, n=30 ($n = |P| + |D|$).

				SB		GMO		TCNE		BBNE		TINE	
instance	$\|P\|$	$\|D\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
E8_21_50_20	6	24	341	—	0.55	—	1.92	—	1.31	—	3.63	—	0.14
E8_21_50_30	9	21	341	—	1.45	—	2.29	—	1.48	—	2.58	—	0.08
E8_21_50_40	12	18	341	—	0.71	—	1.67	—	1.67	—	2.60	—	0.09
E10_21_50_20	6	24	351	—	2.82	—	0.49	—	2.01	—	1.96	—	0.09
E10_21_50_30	9	21	351	—	1.31	—	0.18	—	1.53	—	1.48	—	0.03
E10_21_50_40	12	18	351	—	2.06	—	0.18	—	1.78	—	2.61	—	0.03
E10_51_80_20	6	24	318	—	1.06	—	0.35	—	1.08	—	1.03	—	0.04
E10_51_80_30	9	21	318	—	2.16	—	0.35	—	0.75	—	1.11	—	0.07
E10_51_80_40	12	18	318	—	1.25	—	0.49	—	1.82	—	1.04	—	0.07
F10_15_44_20	6	24	519	—	2839.29	—	0.15	—	8.03	—	2.29	—	0.03
F10_15_44_30	9	21	519	—	2319.88	—	0.14	—	6.64	—	1.72	—	0.04
F10_15_44_40	12	18	519	—	2210.41	—	0.15	—	3.35	—	3.68	—	0.06
F12_21_50_20	6	24	154	—	1.01	—	0.12	—	0.64	—	2.78	—	0.08
F12_21_50_30	9	21	154	—	0.62	—	0.16	—	0.83	—	0.57	—	0.06
F12_21_50_40	12	18	154	—	0.67	—	0.16	—	0.87	—	1.20	—	0.03
F12_51_80_20	6	24	244	—	87.58	—	0.19	—	2.64	—	3.96	—	0.07
F12_51_80_30	9	21	244	—	109.38	—	0.25	—	2.29	—	3.26	—	0.06
F12_51_80_40	12	18	249	1.86	t.lim.	—	0.77	—	3.02	—	18.78	—	0.12

Table 2.13: Detailed results for Table 2.1: GMO set, $25 \leq n \leq 30$ ($n = |P| + |D|$).

				SB		GMO		TCNE		BBNE		TINE	
instance	$\|P\|$	$\|D\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
A_30_L_400	8	17	11376373	—	0.74	—	1	—	1.03	—	1.66	—	0.06
A_30_L_500	8	17	11458360	—	0.99	—	1	—	0.83	—	1.39	—	0.05
A_30_L_1700	8	17	11936711	—	0.67	—	1	—	0.72	—	0.49	—	0.02
A_50_L_500	13	12	10886313	—	2.44	—	1	—	0.82	—	1.89	—	0.03
A_50_L_1000	13	12	11558540	—	2.20	—	1	—	0.86	—	2.27	—	0.05
A_50_L_1700	13	12	12138754	—	4.56	—	1	—	0.92	—	5.76	—	0.11
A_O_L_200	5	20	11193089	—	0.83	—	1	—	0.91	—	0.86	—	0.05
A_O_L_400	5	20	11444625	—	0.79	—	1	—	1.00	—	1.12	—	0.01
A_O_L_1700	5	20	11936711	—	0.56	—	1	—	0.65	—	1.13	—	0.06
B_30_L_100	9	21	11598071	—	7.80	—	1	—	1.70	—	10.18	—	0.07
B_30_L_700	9	21	12403680	—	3.15	—	3	—	2.64	—	2.70	—	0.27
B_30_L_1000	9	21	12437396	—	1.51	—	3	—	1.63	—	2.25	—	0.24
B_50_L_100	15	15	11514188	—	166.50	—	1	—	1.77	—	14.89	—	0.18
B_50_L_700	15	15	12384340	—	14.49	—	4	—	2.99	—	12.04	—	0.37
B_50_L_1000	15	15	12617740	—	19.41	—	7	—	2.83	—	19.19	—	0.27
B_O_L_100	10	20	11598071	—	15.14	—	1	—	2.62	—	4.70	—	0.04
B_O_L_700	10	20	12403680	—	3.57	—	4	—	1.94	—	2.05	—	0.30
B_O_L_1000	10	20	12437396	—	3.19	—	7	—	1.54	—	1.88	—	0.10

Table 2.14: Detailed results for Table 2.1: GMO set, $38 \leq n \leq 60$ ($n = |P| + |D|$).

				SB		GMO		TCNE		BBNE		TINE	
instance	$\|P\|$	$\|D\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
C_30_L_100	12	28	12954837	—	5.51	—	2	—	4.34	—	16.50	—	0.13
C_30_L_200	12	28	13419793	—	7.11	—	1	—	4.26	—	13.97	—	0.21
C_30_L_500	12	28	14062470	—	10.35	—	2	—	9.24	—	7.14	—	0.40
C_50_O_L_100	20	20	11658312	—	2.30	—	1	—	3.06	—	12.87	—	0.18
C_50_O_L_200	20	20	12471864	—	2.60	—	1	—	2.38	—	12.15	—	0.15
C_50_O_L_500	20	20	13814276	—	27.56	—	1	—	10.48	—	19.82	—	0.14
D_30_L_400	11	27	12084482	—	512.81	—	1	—	1.91	—	9.65	—	0.08
D_30_L_700	11	27	12400614	—	438.44	—	1	—	2.42	—	7.87	—	0.15
D_30_L_1200	11	27	12606406	—	705.59	—	1	—	3.31	—	9.55	—	0.19
D_50_L_400	19	19	12290137	3.92	t.lim.	—	6	—	6.20	—	38.65	—	0.41
D_50_L_500	19	19	12480072	4.14	t.lim.	—	12	—	3.58	—	74.13	—	0.35
D_50_L_700	19	19	12706204	3.72	t.lim.	—	15	—	10.09	—	99.21	—	0.39
D_O_L_500	8	30	12239682	—	610.17	—	1	—	2.58	—	8.90	—	0.15
D_O_L_1000	8	30	12579714	—	810.94	—	1	—	2.84	—	7.45	—	0.15
D_O_L_1500	8	30	12606406	—	413.95	—	1	—	1.70	—	3.73	—	0.17
E_30_L_300	14	31	14165760	—	785.91	—	5	—	8.99	—	56.41	—	0.91
E_30_L_500	14	31	14978047	—	816.08	—	4	—	25.46	—	32.45	—	0.65
E_30_L_1200	14	31	15607486	—	129.83	—	5	—	27.67	—	32.31	—	1.94
E_50_L_300	23	22	13219046	—	670.92	—	2	—	21.59	—	46.67	—	0.81
E_50_L_600	23	22	14912502	—	1114.21	—	8	—	36.32	—	138.90	—	1.76
E_50_L_1300	23	22	15882369	—	2754.89	—	226	—	63.16	—	486.07	—	18.00
E_O_L_100	15	30	13262096	—	753.10	—	5	—	17.67	—	94.90	—	1.09
E_O_L_400	15	30	14604360	—	479.45	—	7	—	9.75	—	44.36	—	0.70
E_O_L_900	15	30	15512086	—	149.00	—	9	—	30.55	—	29.95	—	1.00
F_30_L_200	18	42	15426659	2.14	t.lim.	—	11	—	90.45	—	1198.49	—	2.65
F_30_L_400	18	42	16253023	3.23	t.lim.	—	119	—	96.62	—	408.15	—	2.99
F_30_L_1700	18	42	17528624	4.73	t.lim.	—	43	—	92.70	—	657.11	—	2.89
F_50_O_L_200	30	30	15003977	3.27	t.lim.	—	84	—	113.95	7.64	t.lim.	—	10.48
F_50_O_L_400	30	30	16171570	5.15	t.lim.	—	521	—	139.80	5.38	t.lim.	—	32.66
F_50_O_L_1000	30	30	17317966	2.60	t.lim.	—	202	—	142.90	7.65	t.lim.	—	11.70
G_30_L_300	17	40	15758461	4.81	t.lim.	—	2	—	62.44	—	425.65	—	1.42
G_30_L_500	17	40	15978604	—	2492.74	—	2	—	60.74	—	208.40	—	0.72
G_30_L_2000	17	40	16523165	—	822.75	—	36	—	60.90	—	127.38	—	1.63
G_50_L_300	29	28	14800857	—	1468.31	—	19	—	77.05	—	1036.83	—	2.19
G_50_L_500	29	28	15740523	2.05	t.lim.	—	84	—	133.66	—	788.14	—	6.84
G_50_L_1000	29	28	16545438	1.76	t.lim.	—	234	—	135.07	—	583.93	—	29.05
G_O_L_300	12	45	16047591	5.14	t.lim.	—	24	—	59.22	—	291.54	—	1.35
G_O_L_500	12	45	16394865	—	1577.59	—	79	—	79.45	—	247.52	—	1.26
G_O_L_1000	12	45	16523165	—	1621.00	—	56	—	65.14	—	113.80	—	2.38

Table 2.15: Detailed results for Table 2.1: GMO set, $68 \leq n \leq 90$ ($n = |P| + |D|$).

				SB		GMO		TCNE		BBNE		TINE	
instance	$\|P\|$	$\|D\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
H_30_L_300	20	48	17147339	0.49	t.lim.	—	5	—	168.63	—	515.32	—	1.32
H_30_L_500	20	48	17327696	—	1752.99	—	7	—	133.67	—	247.99	—	1.61
H_30_L_1000	20	48	17480958	—	1636.56	—	10	—	52.46	—	352.19	—	0.83
H_50_L_200	34	34	16362687	5.89	t.lim.	—	29	—	215.50	5.69	t.lim.	—	7.73
H_50_L_300	34	34	16843003	3.77	t.lim.	—	18	—	175.23	0.20	t.lim.	—	2.17
H_50_L_400	34	34	17140925	2.07	t.lim.	—	72	—	193.27	3.60	t.lim.	—	5.67
H_50_L_600	34	34	17489325	2.21	t.lim.	—	662	—	322.38	3.60	t.lim.	—	74.05
H_50_L_700	34	34	17663525	2.65	t.lim.	—	13855	—	352.17	4.42	t.lim.	—	55.04
H_O_L_300	23	45	17062720	2.01	t.lim.	—	1	—	122.93	—	1122.47	—	0.86
H_O_L_500	23	45	17327696	0.37	t.lim.	—	4	—	113.91	—	580.77	—	1.23
H_O_L_1000	23	45	17480958	0.76	t.lim.	—	13	—	142.20	—	278.36	—	0.93
I_30_L_100	27	63	18132091	2.66	t.lim.	—	169	—	588.17	6.67	t.lim.	—	9.45
I_30_L_200	27	63	18721168	2.98	t.lim.	—	261	—	319.47	5.97	t.lim.	—	30.89
I_30_L_300	27	63	19159066	2.99	t.lim.	—	3150	—	342.42	7.14	t.lim.	—	29.05
I_50_O_L_100	45	45	17112971	6.44	t.lim.	—	1546	—	718.07	12.13	t.lim.	—	149.22
I_50_O_L_200	45	45	18218462	5.38	t.lim.	—	18880	—	584.34	11.48	t.lim.	—	120.90
I_50_O_L_300	45	45	18989572	4.94	t.lim.	—	21001	—	803.12	11.53	t.lim.	—	143.10

Table 2.16: Detailed results for Table 2.2: GLS set, $15 \leq n \leq 30$ ($n = |PD|$).

			GLS		SB		TCEE		TA		2S		MEN	
instance	$\|PD\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
016_B_p_two	15	221.00	—	423.4	—	377.4	—	4.9	—	0.1	—	0.1	—	0.0
016_B_half	15	243.07	7.8	t.lim.	—	575.1	—	12.1	—	0.0	—	0.1	—	0.0
016_B_one	15	262.20	16.1	t.lim.	—	916.0	—	13.4	—	0.2	—	0.3	—	0.0
016_B_two	15	289.73	23.6	t.lim.	—	832.9	—	14.5	—	0.1	—	0.3	—	0.0
021_B_p_two	20	265.04	0.5	t.lim.	1.8	t.lim.	—	18.0	—	0.1	—	0.0	—	0.0
021_B_half	20	292.18	9.3	t.lim.	—	87.8	—	19.3	—	0.0	—	0.2	—	0.0
021_B_one	20	316.89	17.5	t.lim.	2.6	t.lim.	—	39.3	—	0.0	—	0.1	—	0.0
021_B_two	20	348.13	24.6	t.lim.	2.4	t.lim.	—	24.0	—	0.1	—	0.1	—	0.0
022_B_p_two	21	287.91	8.7	t.lim.	6.6	t.lim.	—	470.2	—	1.6	—	0.5	—	0.2
022_B_half	21	319.76	22.0	t.lim.	9.5	t.lim.	—	314.7	—	0.6	—	0.9	—	0.1
022_B_one	21	343.74	26.8	t.lim.	4.4	t.lim.	—	580.2	—	0.9	—	0.7	—	0.2
022_B_two	21	380.16	33.2	t.lim.	4.1	t.lim.	—	253.7	—	0.4	—	0.2	—	0.2
023_B_p_two	22	527.71	20.1	t.lim.	11.2	t.lim.	—	564.6	—	0.8	—	0.7	—	0.5
023_B_half	22	565.80	20.5	t.lim.	7.5	t.lim.	—	80.7	—	0.1	—	0.4	—	0.1
023_B_one	22	595.20	32.0	t.lim.	7.0	t.lim.	—	63.0	—	0.1	—	0.3	—	0.1
023_B_two	22	653.99	55.3	t.lim.	5.0	t.lim.	—	37.2	—	0.2	—	0.3	—	0.1
026_B_p_two	25	319.54	10.7	t.lim.	10.2	t.lim.	—	117.7	—	0.6	—	0.6	—	0.1
026_B_half	25	343.58	16.6	t.lim.	10.5	t.lim.	—	534.7	—	0.6	—	0.2	—	0.2
026_B_one	25	374.79	30.1	t.lim.	6.1	t.lim.	—	288.8	—	4.0	—	2.0	—	0.3
026_B_two	25	409.67	33.7	t.lim.	5.9	t.lim.	—	871.7	—	0.2	—	0.6	—	0.3
030_B_p_two	29	402.70	25.6	t.lim.	22.7	t.lim.	5.1	t.lim.	—	1.0	—	0.7	—	0.3
030_B_half	29	422.47	31.9	t.lim.	19.8	t.lim.	3.8	t.lim.	—	0.8	—	0.5	—	0.3
030_B_one	29	450.10	39.1	t.lim.	18.9	t.lim.	3.1	t.lim.	—	3.2	—	0.7	—	0.3
030_B_two	29	505.34	48.3	t.lim.	16.2	t.lim.	3.1	t.lim.	—	10.8	—	0.3	—	0.2
031_B_p_two	30	326.60	6.3	t.lim.	4.1	t.lim.	—	288.3	—	2.8	—	1.8	—	0.5
031_B_half	30	361.23	20.7	t.lim.	2.7	t.lim.	—	1256.8	—	1.3	—	1.3	—	0.4
031_B_one	30	393.78	26.9	t.lim.	3.0	t.lim.	—	1750.9	—	1.0	—	1.4	—	0.8
031_B_two	30	454.10	46.6	t.lim.	2.4	t.lim.	0.4	t.lim.	—	15.1	—	2.5	—	0.5

Table 2.17: Detailed results for Table 2.2: GLS set, $32 \leq n \leq 50$ ($n = |PD|$).

			GLS		SB		TCEE		TA		2S		MEN	
instance	$\|PD\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
033_B_p_two	32	459.03	24.0	t.lim.	19.1	t.lim.	3.4	t.lim.	—	2.4	—	3.3	—	1.6
033_B_half	32	490.40	27.4	t.lim.	16.7	t.lim.	2.6	t.lim.	—	1.1	—	1.5	—	1.6
033_B_one	32	516.49	30.4	t.lim.	17.0	t.lim.	2.4	t.lim.	—	2.9	—	0.7	—	0.3
033_B_two	32	556.56	35.5	t.lim.	15.6	t.lim.	0.7	t.lim.	—	5.0	—	0.6	—	0.6
036_B_p_two	35	360.39	10.3	t.lim.	7.8	t.lim.	0.2	t.lim.	—	2.3	—	4.0	—	1.8
036_B_half	35	397.93	19.9	t.lim.	5.3	t.lim.	0.5	t.lim.	—	1.0	—	0.5	—	0.8
036_B_one	35	428.00	36.9	t.lim.	3.9	t.lim.	0.9	t.lim.	—	1.6	—	1.2	—	0.4
036_B_two	35	488.13	45.8	t.lim.	3.2	t.lim.	—	3406.7	—	1.2	—	1.6	—	0.6
041_B_p_two	40	374.70	10.0	t.lim.	5.7	t.lim.	0.2	t.lim.	—	315.4	—	258.4	—	8.4
041_B_half	40	426.53	32.7	t.lim.	3.2	t.lim.	0.4	t.lim.	—	1.0	—	2.3	—	0.5
041_B_one	40	463.90	34.0	t.lim.	3.9	t.lim.	1.6	t.lim.	—	7.0	—	5.6	—	4.0
041_B_two	40	525.12	41.5	t.lim.	3.7	t.lim.	1.6	t.lim.	—	16.3	—	4.9	—	2.8
045_B_p_two	44	671.79	69.4	t.lim.	30.7	t.lim.	10.0	t.lim.	—	0.8	—	0.7	—	0.4
045_B_half	44	693.18	73.9	t.lim.	28.8	t.lim.	6.2	t.lim.	—	0.3	—	0.7	—	0.1
045_B_one	44	720.09	44.4	t.lim.	27.6	t.lim.	3.9	t.lim.	—	0.4	—	0.5	—	0.2
045_B_two	44	773.92	77.5	t.lim.	25.3	t.lim.	4.8	t.lim.	—	0.3	—	0.5	—	0.1
048_B_p_two	47	36703.71	28.4	t.lim.	16.1	t.lim.	5.7	t.lim.	0.1	t.lim.	0.1	t.lim.	—	3.2
048_B_half	47	39983.45	33.1	t.lim.	16.3	t.lim.	4.1	t.lim.	<0.1	t.lim.	<0.1	t.lim.	—	2.7
048_B_one	47	47319.14	43.2	t.lim.	13.8	t.lim.	4.1	t.lim.	—	25.5	—	1.1	—	1.3
048_B_two	47	60457.70	56.7	t.lim.	10.1	t.lim.	2.2	t.lim.	—	3.4	—	1.2	—	0.8
051_B_p_two	50	447.85	28.1	t.lim.	10.8	t.lim.	2.1	t.lim.	0.3	t.lim.	0.3	t.lim.	—	36.7
051_B_half	50	498.17	25.5	t.lim.	10.1	t.lim.	3.3	t.lim.	0.7	t.lim.	0.7	t.lim.	—	32.5
051_B_one	50	530.96	29.4	t.lim.	8.3	t.lim.	3.9	t.lim.	0.7	t.lim.	0.8	t.lim.	—	26.7
051_B_two	50	595.80	60.8	t.lim.	7.7	t.lim.	4.6	t.lim.	0.7	t.lim.	0.8	t.lim.	—	28.4

Table 2.18: Detailed results for Table 2.2: GLS set, $71 \leq n \leq 100$ ($n = |PD|$).

			GLS		SB		TCEE		TA		2S		MEN	
instance	$\|PD\|$	z_{opt}	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec	gap	sec
072_B_p_two	71	212.49	64.5	t.lim.	28.4	t.lim.	26.7	t.lim.	—	356.2	—	12.3	—	3.6
072_B_half	71	217.78	43.7	t.lim.	27.8	t.lim.	18.4	t.lim.	—	567.9	—	6.4	—	2.9
072_B_one	71	226.61	45.8	t.lim.	19.0	t.lim.	25.8	t.lim.	—	499.3	—	9.6	—	2.5
072_B_two	71	244.25	49.8	t.lim.	105.5	t.lim.	22.7	t.lim.	—	8.5	—	6.9	—	5.1
076_B_p_two	75	560.56	48.0	t.lim.	9.5	t.lim.	4.1	t.lim.	1.1	t.lim.	1.0	t.lim.	—	354.6
076_B_half	75	666.86	68.1	t.lim.	2.1	t.lim.	4.5	t.lim.	—	10.4	—	48.0	—	14.0
076_B_one	75	733.30	71.2	t.lim.	11.2	t.lim.	5.3	t.lim.	—	60.9	—	27.3	—	20.6
076_B_two	75	866.22	50.3	t.lim.	31.3	t.lim.	5.8	t.lim.	—	4.5	—	10.5	—	2.7
101_B_p_two	100	658.66	37.2	m.lim.	8.7	t.lim.	4.8	t.lim.	0.5	t.lim.	0.5	t.lim.	—	514.3
101_B_half_a	100	780.00	59.8	m.lim.	0.6	t.lim.	4.5	t.lim.	—	2071.5	—	220.2	—	40.3
101_B_one_a	100	847.91	48.8	m.lim.	8.5	t.lim.	5.7	t.lim.	0.5	t.lim.	—	115.8	—	63.5
101_B_two_a	100	983.71	56.5	m.lim.	63.6	t.lim.	5.2	t.lim.	0.3	t.lim.	—	120.6	—	31.6
101_B_p_two_b	100	536.38	52.1	t.lim.	23.2	t.lim.	26.2	t.lim.	1.0	t.lim.	1.0	t.lim.	0.2	t.lim.
101_B_half_b	100	589.26	53.7	t.lim.	22.9	t.lim.	27.2	t.lim.	0.8	t.lim.	0.8	t.lim.	—	547.0
101_B_one_b	100	628.74	57.3	t.lim.	15.9	t.lim.	25.2	t.lim.	0.6	t.lim.	0.9	t.lim.	—	739.9
101_B_two_b	100	707.68	67.0	t.lim.	25.3	t.lim.	24.9	t.lim.	0.7	t.lim.	0.6	t.lim.	—	722.7

2.D.2 Detailed results for the evaluation of the intermediate dropoff relaxation

Table 2.19: Detailed results for Table 2.3: GLS set, $15 \leq n \leq 30$ ($n = |PD|$).

		TINE (for the SVRPDSP-D)								MEN (for the SVRPDSP)							
instance	$\|PD\|$	z_{opt}	gap	sec	gap_r	gap_c	nodes	feas	gap_d	z_{opt}	gap	sec	gap_r	gap_c	nodes	dupl	iter
016_B_p_two	15	221.00	—	0.0	1.0	—	0	1	—	221.00	—	0.0	1.0	—	0	0	1
016_B_half	15	243.07	—	0.0	0.5	—	0	1	—	243.07	—	0.0	0.5	—	0	0	1
016_B_one	15	262.20	—	0.0	2.2	—	0	1	—	262.20	—	0.0	2.2	—	0	0	1
016_B_two	15	289.73	—	0.0	2.0	—	0	1	—	289.73	—	0.0	2.0	—	0	0	1
021_B_p_two	20	265.04	—	0.1	2.0	0.1	8	1	—	265.04	—	0.0	2.0	0.1	8	0	1
021_B_half	20	292.18	—	0.0	—	—	32	1	—	292.18	—	0.0	—	—	31	0	1
021_B_one	20	316.89	—	0.0	2.6	—	0	1	—	316.89	—	0.0	2.6	—	0	0	1
021_B_two	20	348.13	—	0.0	2.4	—	0	1	—	348.13	—	0.0	2.4	—	0	0	1
022_B_p_two	21	287.91	—	0.4	4.7	3.7	646	1	—	287.91	—	0.2	4.7	3.7	482	0	1
022_B_half	21	319.76	—	0.2	2.6	0.0	147	1	—	319.76	—	0.1	2.6	<0.1	158	0	1
022_B_one	21	343.74	—	0.3	4.1	3.0	496	1	—	343.74	—	0.2	4.1	3.0	430	0	1
022_B_two	21	380.16	—	0.2	3.7	2.7	169	1	—	380.16	—	0.2	3.7	2.7	223	0	1
023_B_p_two	22	527.71	—	0.5	3.9	1.5	571	1	—	527.71	—	0.5	3.9	1.5	863	0	1
023_B_half	22	565.80	—	0.1	4.5	—	0	1	—	565.80	—	0.1	4.5	—	0	0	1
023_B_one	22	595.20	—	0.1	4.3	—	7	1	—	595.20	—	0.1	4.3	—	6	0	1
023_B_two	22	653.99	—	0.1	3.9	—	7	1	—	653.99	—	0.1	3.9	—	6	0	1
026_B_p_two	25	319.54	—	0.2	5.6	0.5	74	1	—	319.54	—	0.1	5.6	0.5	35	0	1
026_B_half	25	343.58	—	0.3	3.7	0.0	311	1	—	343.58	—	0.2	3.7	<0.1	244	0	1
026_B_one	25	374.79	—	0.2	5.3	0.9	67	0	—	374.79	—	0.3	5.3	0.9	262	1	2
026_B_two	25	409.67	—	0.1	4.8	0.0	7	0	—	409.67	—	0.3	4.8	<0.1	56	1	2
030_B_p_two	29	402.70	—	0.7	16.7	0.1	318	1	—	402.70	—	0.3	16.7	0.1	340	0	1
030_B_half	29	422.47	—	0.4	16.7	0.3	153	1	—	422.47	—	0.3	16.7	0.3	384	0	1
030_B_one	29	450.10	—	0.3	15.7	0.2	61	1	—	450.10	—	0.3	15.7	0.2	302	0	1
030_B_two	29	505.34	—	0.3	14.0	0.2	91	1	—	505.34	—	0.2	14.0	0.2	127	0	1
031_B_p_two	30	326.60	—	0.5	2.4	1.6	331	1	—	326.60	—	0.5	2.4	1.6	140	1	2
031_B_half	30	361.23	—	0.6	0.8	0.2	593	1	—	361.23	—	0.4	0.8	0.2	471	0	1
031_B_one	30	393.78	—	1.0	1.3	0.8	635	1	—	393.78	—	0.8	1.3	0.8	690	0	1
031_B_two	30	454.10	—	1.8	1.2	0.7	1881	1	—	454.10	—	0.5	1.2	0.7	506	0	1

Table 2.20: Detailed results for Table 2.3: GLS set, $32 \leq n \leq 50$ ($n = |PD|$).

		TINE (for the SVRPDSP-D)								MEN (for the SVRPDSP)							
instance	$\|PD\|$	z_{opt}	gap	sec	gap_r	gap_c	nodes	feas	gap_d	z_{opt}	gap	sec	gap_r	gap_c	nodes	dupl	iter
033_B_p_two	32	459.03	—	1.8	10.9	1.2	2626	1	—	459.03	—	1.6	10.9	1.2	2767	0	1
033_B_half	32	490.40	—	1.2	10.3	—	940	1	—	490.40	—	1.6	10.3	—	2141	0	1
033_B_one	32	516.49	—	1.0	10.9	0.1	893	1	—	516.49	—	0.3	10.9	2.4	165	0	1
033_B_two	32	556.56	—	0.9	10.1	0.1	662	1	—	556.56	—	0.6	10.1	0.1	525	0	1
036_B_p_two	35	360.39	—	2.4	3.1	2.3	1766	1	—	360.39	—	1.8	3.1	2.3	1209	1	2
036_B_half	35	397.93	—	2.6	0.7	0.1	559	1	—	397.93	—	0.8	0.7	0.1	692	0	1
036_B_one	35	428.00	—	2.1	0.7	0.1	221	1	—	428.00	—	0.4	0.7	—	183	0	1
036_B_two	35	488.13	—	0.7	0.6	—	527	1	—	488.13	—	0.6	0.6	—	433	0	1
041_B_p_two	40	374.70	—	3.9	2.8	2.5	2045	1	—	374.70	—	8.4	2.8	2.5	11034	1	2
041_B_half	40	426.53	—	1.3	0.6	0.1	163	1	—	426.53	—	0.5	0.6	0.1	153	0	1
041_B_one	40	463.90	—	5.9	2.0	1.5	5844	1	—	463.90	—	4.0	2.0	1.5	4836	0	1
041_B_two	40	525.12	—	3.9	1.7	1.3	2888	1	—	525.12	—	2.8	1.7	1.3	2789	0	1
045_B_p_two	44	671.79	—	0.8	17.6	0.3	74	1	—	671.79	—	0.4	17.6	0.3	74	0	1
045_B_half	44	693.18	—	0.1	16.8	—	0	1	—	693.18	—	0.1	16.8	—	0	0	1
045_B_one	44	720.09	—	0.2	16.2	—	3	1	—	720.09	—	0.2	16.2	—	3	0	1
045_B_two	44	773.92	—	0.1	15.1	—	1	1	—	773.92	—	0.1	15.1	—	1	0	1
048_B_p_two	47	36703.71	—	5.0	6.2	1.9	1319	1	—	36703.71	—	3.2	6.2	1.9	676	2	3
048_B_half	47	39983.45	—	6.9	5.7	1.9	2251	1	—	39983.45	—	2.7	5.7	1.9	1035	1	2
048_B_one	47	47319.14	—	2.6	3.7	0.2	423	1	—	47319.14	—	1.3	3.7	0.2	515	0	1
048_B_two	47	60457.70	—	5.1	2.9	0.1	1036	1	—	60457.70	—	0.8	2.9	0.1	99	0	1
051_B_p_two	50	447.85	—	31.2	3.0	1.8	23890	1	—	447.85	—	36.7	3.0	1.8	29186	1	2
051_B_half	50	498.17	—	62.1	3.2	1.9	46946	1	—	498.17	—	32.5	3.2	2.1	22605	1	2
051_B_one	50	530.96	—	40.6	3.1	1.9	25665	1	—	530.96	—	26.7	3.1	1.9	22717	1	2
051_B_two	50	595.80	—	68.6	2.7	1.7	42107	1	—	595.80	—	28.4	2.7	1.7	21177	1	2

Table 2.21: Detailed results for Table 2.3: GLS set, $71 \leq n \leq 100$ ($n = |PD|$). The value z_{opt}=536.38 for instance 101_B_p_two_b was obtained by running MEN for 17172 seconds and 2410673 nodes.

		TINE (for the SVRPDSP-D)								MEN (for the SVRPDSP)							
instance	$\|PD\|$	z_{opt}	gap	sec	gap_r	gap_c	nodes	feas	gap_d	z_{opt}	gap	sec	gap_r	gap_c	nodes	dupl	iter
072_B_p_two	71	212.49	—	10.5	17.0	0.6	486	1	—	212.49	—	3.6	17.0	0.6	416	0	1
072_B_half	71	217.78	—	28.5	16.6	0.6	594	1	—	217.78	—	2.9	16.6	0.6	214	0	1
072_B_one	71	226.61	—	6.6	15.9	0.5	68	1	—	226.61	—	2.5	15.9	0.5	174	0	1
072_B_two	71	244.25	—	9.2	14.8	0.5	278	1	—	244.25	—	5.1	14.8	0.5	348	0	1
076_B_p_two	75	559.16	0.25	355.4	1.6	1.4	96245	0	0.2	560.56	—	354.6	1.9	1.6	64000	3	4
076_B_half	75	666.86	—	7.4	0.5	0.1	1099	1	—	666.86	—	14.0	0.5	0.1	2868	0	1
076_B_one	75	733.30	—	39.1	0.5	0.1	9001	1	—	733.30	—	20.6	0.5	0.1	5068	0	1
076_B_two	75	866.22	—	3.2	0.4	0.1	270	1	—	866.22	—	2.7	0.4	0.1	310	0	1
101_B_p_two	100	657.52	0.17	122.1	2.3	1.3	8796	0	0.2	658.66	—	514.3	2.5	1.5	54226	4	5
101_B_half_a	100	780.00	—	82.7	1.6	0.9	14539	1	—	780.00	—	40.3	1.6	0.9	7997	0	1
101_B_one_a	100	847.91	—	160.7	1.5	0.8	13845	1	—	847.91	—	63.5	1.5	0.8	12495	0	1
101_B_two_a	100	983.71	—	145.6	1.3	0.7	9615	1	—	983.71	—	31.6	1.3	0.7	4718	0	1
101_B_p_two_b	100	534.72	0.34	t.lim.	27.4	1.7	335348	0	0.3	536.38	0.2	t.lim.	27.6	2.0	184356	3	4
101_B_half_b	100	589.23	0.01	318.3	25.1	1.5	49924	0	0.0	589.26	—	547.0	25.1	1.5	48811	3	4
101_B_one_b	100	628.71	0.00	302.1	23.3	1.4	42555	0	0.0	628.74	—	739.9	23.3	1.4	59168	3	4
101_B_two_b	100	707.65	0.00	450.8	20.7	1.2	58697	0	0.0	707.68	—	722.7	20.7	1.2	69607	3	4

2.D.3 Detailed results for the case of mandatory pickups

Table 2.22: Detailed results for Table 2.4: GHLV set, $15 \leq n \leq 30$ ($n = |PD|$).

			GHLV		TA-MP		2S-MP		MEN-MP	
instance	$\|PD\|$	z_{opt}	sec	gap	sec	gap	sec	gap	sec	gap
016_B02_CA	15	220.99	t.lim.	2.7	0.0	—	0.0	—	0.1	—
016_B02_CB	15	220.74	1068.5	—	0.1	—	0.1	—	0.0	—
021_B02_CA	20	267.23	t.lim.	4.2	0.7	—	0.6	—	0.6	—
021_B02_CB	20	261.81	t.lim.	5.2	0.5	—	0.7	—	0.4	—
022_B02_CA	21	278.43	t.lim.	13.1	0.3	—	0.1	—	0.1	—
022_B02_CB	21	278.43	t.lim.	14.5	0.1	—	0.5	—	0.5	—
023_B02_CA	22	482.78	t.lim.	9.9	0.2	—	0.5	—	0.1	—
023_B02_CB	22	482.71	t.lim.	16.9	0.1	—	0.4	—	0.1	—
026_B02_CA	25	306.93	t.lim.	18.0	0.2	—	0.1	—	0.2	—
026_B02_CB	25	314.16	t.lim.	15.2	0.5	—	0.1	—	0.6	—
030_B02_CA	29	388.43	t.lim.	25.2	0.5	—	0.5	—	0.7	—
030_B02_CB	29	386.9	t.lim.	18.0	0.5	—	0.4	—	0.4	—
031_B02_CA	30	316.67	t.lim.	3.1	0.4	—	0.4	—	0.2	—
031_B02_CB	30	322.32	t.lim.	5.4	1.2	—	2.7	—	1.6	—

Table 2.23: Detailed results for Table 2.4: GHLV set, $32 \leq n \leq 50$ ($n = |PD|$).

			GHLV		TA-MP		2S-MP		MEN-MP	
instance	$\|PD\|$	z_{opt}	sec	gap	sec	gap	sec	gap	sec	gap
033_B02_CA	32	447.05	t.lim.	20.3	15.9	—	0.5	—	0.8	—
033_B02_CB	32	463.68	t.lim.	17.5	t.lim.	1.1	t.lim.	1.2	1.4	—
036_B02_CA	35	353.26	t.lim.	6.8	1.6	—	0.9	—	1.4	—
036_B02_CB	35	359.54	t.lim.	7.6	10.4	—	1.1	—	1.4	—
041_B02_CA	40	365.6	t.lim.	5.0	1.1	—	0.6	—	0.4	—
041_B02_CB	40	367.97	t.lim.	5.9	5.3	—	2.4	—	2.2	—
045_B02_CA	44	619.17	t.lim.	34.3	0.5	—	0.4	—	0.3	—
045_B02_CB	44	619.18	t.lim.	34.7	0.4	—	0.4	—	0.4	—
048_B02_CA	47	33523.67	t.lim.	20.0	0.7	—	0.5	—	0.8	—
048_B02_CB	47	34056.27	t.lim.	19.1	1.7	—	9.6	—	1.3	—
051_B02_CA	50	428.86	t.lim.	10.9	12.0	—	3.2	—	2.9	—
051_B02_CB	50	435.51	t.lim.	12.6	3067.0	—	2036.7	—	29.7	—

Table 2.24: Detailed results for Table 2.4: GHLV set, $71 \leq n \leq 100$ ($n = |PD|$).

			GHLV		TA-MP		2S-MP		MEN-MP	
instance	$\|PD\|$	z_{opt}	sec	gap	sec	gap	sec	gap	sec	gap
072_B02_CA	71	198.92	t.lim.	33.6	81.8	—	6.8	—	6.7	—
072_B02_CB	71	194.68	t.lim.	32.1	3.2	—	2.3	—	2.3	—
076_B02_CA	75	545.59	t.lim.	10.0	5.8	—	4.6	—	4.4	—
076_B02_CB	75	548.27	t.lim.	10.6	859.5	—	37.9	—	36.2	—
101_B02_CA_a	100	640.13	t.lim.	9.2	7.7	—	8.3	—	8.1	—
101_B02_CB_a	100	646.39	t.lim.	10.1	t.lim.	0.5	1441.0	—	560.9	—
101_B02_CA_b	100	503.96	t.lim.	35.3	60.3	—	61.0	—	58.0	—
101_B02_CB_b	100	503.96	t.lim.	34.9	1344.7	—	29.6	—	29.0	—

2.D.4 Detailed results for the new large size instances

Table 2.25: Detailed results for Table 2.7: selective pickups, BI set ($n = |PD|$).

			MEN			
instance	$\|PD\|$	UB_{best}	UB	sec	gap	nodes
121_B_p_two	120	606.07	607.60	t.lim.	3.3	170600
121_B_half	120	739.39	739.39	t.lim.	0.3	271802
121_B_one	120	851.97	866.29	t.lim.	0.9	113068
121_B_two	120	1060.42	1060.42	2034.4	—	278810
135_B_p_two	134	755.24	785.75	t.lim.	1.5	119824
135_B_half	134	765.09	765.09	t.lim.	1.0	61780
135_B_one	134	795.69	795.69	t.lim.	1.3	135104
135_B_two	134	845.32	845.32	t.lim.	1.2	50265
151_B_p_two	150	755.88	755.88	1335.9	—	81717
151_B_half	150	863.09	863.09	747.7	—	33432
151_B_one	150	946.75	946.75	214.0	—	9810
151_B_two	150	1114.10	1114.10	380.7	—	18055
200_B_p_two	199	860.83	860.83	t.lim.	2.8	70647
200_B_half	199	1021.23	1021.23	3076.3	—	19019
200_B_one	199	1152.87	1152.87	339.6	—	11771
200_B_two	199	1416.95	1416.95	t.lim.	0.1	98431

Table 2.26: Detailed results for Table 2.7: mandatory pickups, BI-MP set ($n = |PD|$).

			MEN-MP			
instance	$\|PD\|$	UB_{best}	UB	sec	gap	nodes
121_B02_CA	120	544.16	544.16	103.6	—	2952
121_B02_CB	120	549.16	549.16	1817.0	—	82068
135_B02_CA	134	720.56	720.56	3026.7	—	14968
135_B02_CB	134	721.85	721.85	1488.5	—	10992
151_B02_CA	150	710.16	710.16	366.9	—	3502
151_B02_CB	150	714.66	714.66	316.2	—	343
200_B02_CA	199	790.5	790.5	t.lim.	1.6	36200
200_B02_CB	199	782.43	782.43	t.lim.	0.5	56000

2.D.5 Detailed results for the case of multiple vehicles

Table 2.27: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.6$, $15 \leq n \leq 30$ ($n = |PD|$)).

		$k=2$				$k=3$				$k=4$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
016_B_p_two	15	238.66	0.0	—	1	251.64	0.0	—	1	269.22	0.1	—	1
016_B_half	15	238.66	0.0	—	1	251.64	0.0	—	1	269.22	0.1	—	1
016_B_one	15	238.66	0.0	—	1	251.64	0.0	—	1	269.22	0.1	—	1
016_B_two	15	238.66	0.0	—	1	251.64	0.0	—	1	269.22	0.1	—	1
021_B_p_two	20	278.40	0.4	—	1	287.04	0.0	—	1	302.51	0.0	—	1
021_B_half	20	289.29	7.7	—	3	287.04	0.0	—	1	302.51	0.0	—	1
021_B_one	20	289.29	3.8	—	3	287.04	0.0	—	1	302.51	0.0	—	1
021_B_two	20	289.29	3.4	—	3	287.04	0.0	—	1	302.51	0.0	—	1
022_B_p_two	21	290.43	0.5	—	1	301.72	0.1	—	1	320.79	0.1	—	1
022_B_half	21	297.82	0.2	—	1	308.21	0.2	—	1	320.79	0.1	—	1
022_B_one	21	297.82	0.3	—	1	308.21	0.2	—	1	320.79	0.1	—	1
022_B_two	21	297.82	0.3	—	1	308.21	0.2	—	1	320.79	0.1	—	1
023_B_p_two	22	547.05	18.6	—	4	550.50	5.7	—	3	583.16	0.8	—	3
023_B_half	22	567.59	19.6	—	4	568.14	9.7	—	3	583.16	0.8	—	3
023_B_one	22	596.99	16.2	—	4	569.20	4.7	—	3	583.16	0.8	—	3
023_B_two	22	655.79	14.2	—	4	569.20	3.1	—	3	583.16	0.8	—	3
026_B_p_two	25	334.00	0.8	—	1	342.75	0.2	—	1	361.86	0.3	—	1
026_B_half	25	337.22	1.9	—	2	342.75	0.2	—	1	361.86	0.3	—	1
026_B_one	25	337.22	1.9	—	2	342.75	0.2	—	1	361.86	0.3	—	1
026_B_two	25	337.22	2.2	—	2	342.75	0.4	—	1	361.86	0.3	—	1
030_B_p_two	29	390.12	0.2	—	1	408.55	0.5	—	1	430.11	0.3	—	1
030_B_half	29	390.12	0.2	—	1	408.55	0.3	—	1	430.11	0.3	—	1
030_B_one	29	390.12	0.2	—	1	408.55	0.5	—	1	430.11	0.3	—	1
030_B_two	29	390.12	0.2	—	1	408.55	0.7	—	1	430.11	0.3	—	1
031_B_p_two	30	331.88	0.9	—	1	342.25	0.8	—	1	353.98	0.5	—	1
031_B_half	30	331.88	0.9	—	1	342.25	0.8	—	1	353.98	0.5	—	1
031_B_one	30	331.88	0.9	—	1	342.25	0.8	—	1	353.98	0.5	—	1
031_B_two	30	331.88	0.9	—	1	342.25	0.8	—	1	353.98	0.5	—	1

Table 2.28: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.6$, $32 \leq n \leq 50$ ($n = |PD|$)).

		$k = 2$				$k = 3$				$k = 4$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
033_B_p_two	32	558.94	13.9	—	3	620.50	1.1	—	1	688.65	0.5	—	1
033_B_half	32	558.94	9.1	—	3	620.50	1.5	—	1	688.65	0.5	—	1
033_B_one	32	558.94	12.1	—	3	620.50	1.8	—	1	688.65	0.5	—	1
033_B_two	32	558.94	12.6	—	3	620.50	1.1	—	1	688.65	0.5	—	1
036_B_p_two	35	360.54	0.3	—	1	370.54	0.9	—	1	380.91	0.4	—	1
036_B_half	35	360.54	0.3	—	1	370.54	0.9	—	1	380.91	0.5	—	1
036_B_one	35	360.54	0.4	—	1	370.54	0.9	—	1	380.91	0.4	—	1
036_B_two	35	360.54	0.4	—	1	370.54	0.9	—	1	380.91	0.5	—	1
041_B_p_two	40	376.64	4.9	—	1	386.86	5.3	—	1	396.48	4.4	—	1
041_B_half	40	377.71	2.7	—	1	386.86	6.4	—	1	397.48	4.0	—	1
041_B_one	40	377.71	2.9	—	1	386.86	6.6	—	1	397.48	3.0	—	1
041_B_two	40	377.71	2.9	—	1	386.86	3.1	—	1	397.48	3.1	—	1
045_B_p_two	44	640.47	692.0	—	5	636.30	0.6	—	1	641.80	2.1	—	1
045_B_half	44	640.47	587.4	—	5	636.30	0.6	—	1	641.80	2.1	—	1
045_B_one	44	640.47	617.3	—	5	636.30	0.6	—	1	641.80	2.1	—	1
045_B_two	44	640.47	700.6	—	6	636.30	0.6	—	1	641.80	2.1	—	1
048_B_p_two	47	34693.66	3.1	—	1	35410.25	4.4	—	1	36254.40	1.5	—	1
048_B_half	47	34693.66	2.2	—	1	35410.25	4.5	—	1	36254.40	1.5	—	1
048_B_one	47	34693.66	2.2	—	1	35410.25	5.9	—	1	36254.40	1.5	—	1
048_B_two	47	34693.66	2.2	—	1	35410.25	6.5	—	1	36254.40	1.5	—	1
051_B_p_two	50	443.99	11.9	—	1	447.19	19.0	—	1	457.89	12.5	—	1
051_B_half	50	444.31	9.3	—	1	448.42	15.4	—	1	458.02	8.3	—	1
051_B_one	50	444.31	19.6	—	1	448.42	24.0	—	1	458.02	8.8	—	1
051_B_two	50	444.31	10.8	—	1	448.42	20.0	—	1	458.02	8.9	—	1

Table 2.29: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.6$, $71 \leq n \leq 100$ ($n = |PD|$)).

		$k = 2$				$k = 3$				$k = 4$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
072_B_p_two	71	204.06	4.4	—	1	215.44	2.1	—	1	227.39	5.0	—	1
072_B_half	71	204.06	4.5	—	1	215.44	2.2	—	1	227.39	5.0	—	1
072_B_one	71	204.06	4.4	—	1	215.44	2.1	—	1	227.39	5.0	—	1
072_B_two	71	204.06	4.3	—	1	215.44	2.1	—	1	227.39	5.0	—	1
076_B_p_two	75	556.59	44.2	—	1	562.40	150.9	—	1	569.15	71.6	—	1
076_B_half	75	556.59	193.4	—	1	562.40	108.9	—	1	570.05	21.3	—	1
076_B_one	75	556.59	42.8	—	1	562.40	114.8	—	1	570.05	42.3	—	1
076_B_two	75	556.59	134.6	—	1	562.40	126.1	—	1	570.05	56.9	—	1
101_B_p_two_a	100	651.35	193.7	—	1	656.10	37.0	—	1	665.44	75.8	—	1
101_B_half_a	100	654.53	756.9	—	1	660.66	515.8	—	1	668.23	98.0	—	1
101_B_one_a	100	654.53	848.0	—	1	660.66	776.7	—	1	668.23	156.6	—	1
101_B_two_a	100	654.53	860.6	—	1	660.66	454.9	—	1	668.23	292.0	—	1
101_B_p_two_b	100	526.01	385.5	—	1	544.21	298.7	—	1	564.58	353.6	—	1
101_B_half_b	100	526.01	344.3	—	1	544.21	201.0	—	1	564.58	236.6	—	1
101_B_one_b	100	526.01	313.6	—	1	544.21	532.3	—	1	564.58	229.5	—	1
101_B_two_b	100	526.01	610.0	—	1	544.21	335.6	—	1	564.58	326.5	—	1

Table 2.30: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.5$, $15 \leq n \leq 30$ ($n = |PD|$)).

		$k = 3$				$k = 4$				$k = 5$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
016_B_p_two	15	253.76	0.7	—	1	269.54	0.1	—	1	290.87	0.0	—	1
016_B_half	15	253.76	0.1	—	1	269.54	0.1	—	1	290.87	0.0	—	1
016_B_one	15	253.76	0.1	—	1	269.54	0.1	—	1	290.87	0.0	—	1
016_B_two	15	253.76	0.1	—	1	269.54	0.1	—	1	290.87	0.0	—	1
021_B_p_two	20	297.13	4.0	—	2	304.21	0.0	—	1	326.57	0.0	—	1
021_B_half	20	303.18	1.9	—	1	304.21	0.0	—	1	326.57	0.0	—	1
021_B_one	20	303.18	1.5	—	1	304.21	0.0	—	1	326.57	0.0	—	1
021_B_two	20	303.18	1.2	—	1	304.21	0.0	—	1	326.57	0.0	—	1
022_B_p_two	21	312.00	0.8	—	1	324.76	0.8	—	1	341.15	0.2	—	1
022_B_half	21	318.18	1.3	—	1	329.62	1.0	—	1	341.15	0.1	—	1
022_B_one	21	318.18	0.8	—	1	329.62	0.7	—	1	341.15	0.1	—	1
022_B_two	21	318.18	0.9	—	1	329.62	0.8	—	1	341.15	0.1	—	1
023_B_p_two	22	550.50	5.6	—	3	583.16	0.8	—	3	624.98	0.2	—	1
023_B_half	22	568.14	9.6	—	3	583.16	0.8	—	3	624.98	0.2	—	1
023_B_one	22	569.20	4.6	—	3	583.16	0.8	—	3	624.98	0.2	—	1
023_B_two	22	569.20	3.1	—	3	583.16	0.8	—	3	624.98	0.2	—	1
026_B_p_two	25	359.67	30.9	—	4	367.80	1.4	—	2	386.91	0.1	—	1
026_B_half	25	359.67	6.5	—	2	367.80	2.0	—	2	386.91	0.1	—	1
026_B_one	25	359.67	4.9	—	2	367.80	1.3	—	2	386.91	0.1	—	1
026_B_two	25	359.67	3.9	—	2	367.80	1.7	—	2	386.91	0.1	—	1
030_B_p_two	29	489.59	7.0	—	1	499.22	0.3	—	1	517.65	0.1	—	1
030_B_half	29	493.68	6.2	—	1	499.22	0.3	—	1	517.65	0.1	—	1
030_B_one	29	493.68	3.6	—	1	499.22	0.3	—	1	517.65	0.1	—	1
030_B_two	29	493.68	1.9	—	1	499.22	0.3	—	1	517.65	0.1	—	1
031_B_p_two	30	347.69	15.0	—	1	354.03	1.5	—	1	366.89	1.3	—	1
031_B_half	30	350.64	8.0	—	1	357.66	5.3	—	1	368.38	1.0	—	1
031_B_one	30	350.64	12.6	—	1	358.86	5.4	—	1	368.38	1.4	—	1
031_B_two	30	350.64	4.4	—	1	358.86	3.9	—	1	368.38	1.0	—	1

Table 2.31: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.5$, $32 \leq n \leq 50$ ($n = |PD|$)).

		$k=3$				$k=4$				$k=5$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
033_B_p_two	32	638.53	342.6	—	4	700.64	21.0	—	1	804.60	7.9	—	1
033_B_half	32	668.55	t.lim.	1.67	6	716.36	130.1	—	3	807.83	71.3	—	4
033_B_one	32	675.58	1095.0	—	7	736.40	271.4	—	3	807.83	70.2	—	4
033_B_two	32	675.58	68.4	—	5	739.68	114.9	—	4	807.83	61.4	—	4
036_B_p_two	35	379.92	69.4	—	1	383.65	3.6	—	1	393.85	2.0	—	1
036_B_half	35	382.62	31.4	—	1	388.26	18.0	—	1	397.25	2.4	—	1
036_B_one	35	382.62	9.8	—	1	389.96	13.2	—	1	397.25	2.1	—	1
036_B_two	35	382.62	10.5	—	1	389.96	11.4	—	1	397.25	2.1	—	1
041_B_p_two	40	395.57	216.1	—	1	402.43	72.1	—	1	413.29	14.9	—	1
041_B_half	40	397.59	75.4	—	1	407.90	219.9	—	1	415.10	18.4	—	1
041_B_one	40	397.59	59.8	—	1	407.90	87.0	—	1	415.10	22.3	—	1
041_B_two	40	397.59	39.9	—	1	407.90	85.6	—	1	415.10	18.2	—	1
045_B_p_two	44	643.48	10.4	—	2	643.48	25.9	—	2	648.06	0.5	—	1
045_B_half	44	643.48	12.7	—	2	643.48	8.8	—	2	648.06	0.5	—	1
045_B_one	44	643.48	37.6	—	3	643.48	15.8	—	2	648.06	0.5	—	1
045_B_two	44	643.48	26.8	—	2	643.48	13.4	—	2	648.06	0.5	—	1
048_B_p_two	47	36552.49	240.2	—	5	37290.99	123.1	—	5	38036.39	15.0	—	1
048_B_half	47	36552.49	126.2	—	5	37290.99	98.6	—	5	38036.39	7.1	—	1
048_B_one	47	36552.49	173.3	—	5	37290.99	121.9	—	5	38036.39	6.5	—	1
048_B_two	47	36552.49	185.0	—	5	37290.99	131.1	—	5	38036.39	6.4	—	1
051_B_p_two	50	455.40	33.6	—	1	459.48	10.2	—	1	474.05	44.2	—	1
051_B_half	50	455.40	32.3	—	1	459.48	5.4	—	1	474.05	32.0	—	1
051_B_one	50	455.40	51.5	—	1	459.48	8.4	—	1	474.05	26.2	—	1
051_B_two	50	455.40	16.4	—	1	459.48	13.2	—	1	474.05	16.0	—	1

Table 2.32: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.5$, $71 \leq n \leq 100$ ($n = |PD|$)).

		$k=3$				$k=4$				$k=5$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
072_B_p_two	71	220.37	35.0	—	1	231.75	92.8	—	1	243.70	18.7	—	1
072_B_half	71	220.37	23.9	—	1	231.75	32.5	—	1	243.70	14.1	—	1
072_B_one	71	220.37	46.3	—	1	231.75	90.1	—	1	243.70	11.3	—	1
072_B_two	71	220.37	21.5	—	1	231.75	77.4	—	1	243.70	16.2	—	1
076_B_p_two	75	571.03	3397.4	—	1	576.01	1314.7	—	1	584.32	2451.0	—	1
076_B_half	75	571.03	t.lim.	0.56	1	576.84	1455.6	—	1	585.44	2601.7	—	1
076_B_one	75	571.03	522.8	—	1	576.84	410.2	—	1	585.44	590.7	—	1
076_B_two	75	571.03	477.3	—	1	576.84	418.7	—	1	585.44	660.3	—	1
101_B_p_two_a	100	661.69	507.1	—	1	670.12	97.7	—	1	679.56	428.2	—	1
101_B_half_a	100	661.69	308.8	—	1	670.12	187.4	—	1	679.56	302.0	—	1
101_B_one_a	100	661.69	399.4	—	1	670.12	274.3	—	1	679.56	441.1	—	1
101_B_two_a	100	661.69	452.1	—	1	670.12	187.1	—	1	679.56	163.3	—	1
101_B_p_two_b	100	553.34	t.lim.	0.73	1	571.84	t.lim.	0.81	1	592.21	t.lim.	0.18	2
101_B_half_b	100	553.23	t.lim.	0.91	1	571.92	t.lim.	0.48	1	592.21	3435.7	—	2
101_B_one_b	100	553.26	t.lim.	0.36	2	573.21	t.lim.	0.74	1	592.21	3206.9	—	2
101_B_two_b	100	553.23	t.lim.	0.41	2	572.29	t.lim.	0.26	2	592.21	3065.0	—	2

Table 2.33: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.4$, $15 \leq n \leq 30$ ($n = |PD|$)).

		$k=3$				$k=4$				$k=5$				$k=6$			
instance	$\|PD\|$	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
016_B_p_two	15	263.16	1.7	—	2	272.95	0.1	—	1	290.87	0.1	—	1	313.41	0.0	—	1
016_B_half	15	268.54	2.9	—	3	272.95	0.1	—	1	290.87	0.1	—	1	313.41	0.0	—	1
016_B_one	15	268.54	2.4	—	3	272.95	0.1	—	1	290.87	0.1	—	1	313.41	0.0	—	1
016_B_two	15	268.54	3.0	—	3	272.95	0.1	—	1	290.87	0.1	—	1	313.41	0.0	—	1
021_B_p_two	20	305.77	0.9	—	1	316.49	0.9	—	2	329.00	0.4	—	1	353.06	0.1	—	1
021_B_half	20	315.95	190.1	—	6	320.22	0.3	—	1	329.00	0.3	—	1	353.06	0.1	—	1
021_B_one	20	315.95	90.5	—	6	320.22	0.3	—	1	329.00	0.6	—	1	353.06	0.1	—	1
021_B_two	20	315.95	53.1	—	6	320.22	0.3	—	1	329.00	0.4	—	1	353.06	0.1	—	1
022_B_p_two	21	340.71	2.5	—	2	345.64	1.9	—	2	357.10	0.6	—	2	369.62	0.0	—	1
022_B_half	21	340.71	4.3	—	2	346.63	0.9	—	1	358.09	0.3	—	1	369.62	0.0	—	1
022_B_one	21	340.71	3.7	—	1	346.63	0.6	—	1	358.09	0.3	—	1	369.62	0.0	—	1
022_B_two	21	340.71	2.0	—	1	346.63	0.5	—	1	358.09	0.2	—	1	369.62	0.0	—	1
023_B_p_two	22	550.50	5.8	—	3	583.16	0.8	—	3	624.98	0.2	—	1	668.46	0.1	—	1
023_B_half	22	568.14	9.7	—	3	583.16	0.8	—	3	624.98	0.2	—	1	668.46	0.1	—	1
023_B_one	22	569.20	4.7	—	3	583.16	0.8	—	3	624.98	0.2	—	1	668.46	0.1	—	1
023_B_two	22	569.20	3.1	—	3	583.16	0.8	—	3	624.98	0.2	—	1	668.46	0.1	—	1
026_B_p_two	25	362.31	1.1	—	1	375.24	0.2	—	1	394.35	0.9	—	1	421.02	0.5	—	1
026_B_half	25	365.49	1.3	—	1	375.24	0.2	—	1	394.35	1.5	—	1	421.02	0.8	—	1
026_B_one	25	365.49	0.9	—	1	375.24	0.3	—	1	394.35	0.9	—	1	421.02	0.7	—	1
026_B_two	25	365.49	1.0	—	1	375.24	0.3	—	1	394.35	1.3	—	1	421.02	0.7	—	1
030_B_p_two	29	493.68	2.2	—	1	505.00	0.3	—	1	523.43	0.3	—	1	544.99	0.3	—	1
030_B_half	29	493.68	4.2	—	1	505.00	0.3	—	1	523.43	0.3	—	1	544.99	0.3	—	1
030_B_one	29	493.68	1.8	—	1	505.00	0.3	—	1	523.43	0.3	—	1	544.99	0.3	—	1
030_B_two	29	493.68	1.5	—	1	505.00	0.3	—	1	523.43	0.3	—	1	544.99	0.3	—	1
031_B_p_two	30	356.27	14.0	—	2	366.64	4.8	—	2	379.10	6.0	—	1	391.96	2.7	—	1
031_B_half	30	356.27	12.8	—	2	366.64	6.2	—	2	379.50	4.3	—	1	392.35	2.3	—	1
031_B_one	30	356.27	14.7	—	2	366.64	9.1	—	2	379.50	4.5	—	1	392.35	1.8	—	1
031_B_two	30	356.27	9.9	—	2	366.64	5.9	—	2	379.50	7.3	—	1	392.35	1.9	—	1

Table 2.34: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.4$, $32 \leq n \leq 50$ ($n = |PD|$)).

		$k=3$				$k=4$				$k=5$				$k=6$			
instance	|PD|	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter	UB	sec	gap	iter
033_B_p_two	32	683.70	17.6	—	1	751.18	8.9	—	1	822.72	6.6	—	1	928.22	7.0	—	1
033_B_half	32	684.81	28.3	—	2	751.18	4.7	—	1	822.72	6.7	—	1	928.22	5.0	—	1
033_B_one	32	684.81	81.0	—	2	751.18	5.5	—	1	822.72	7.7	—	1	928.22	6.3	—	1
033_B_two	32	684.81	28.3	—	2	751.18	8.6	—	1	822.72	3.4	—	1	928.22	3.4	—	1
036_B_p_two	35	387.94	58.5	—	1	396.00	18.7	—	1	408.17	6.4	—	1	421.03	3.2	—	1
036_B_half	35	390.55	203.0	—	2	396.00	13.8	—	1	408.17	8.3	—	1	421.03	3.8	—	1
036_B_one	35	390.55	123.1	—	2	396.00	19.5	—	1	408.17	5.3	—	1	421.03	4.4	—	1
036_B_two	35	390.55	103.9	—	2	396.00	16.9	—	1	408.17	5.1	—	1	421.03	4.6	—	1
041_B_p_two	40	401.88	138.0	—	1	411.73	49.9	—	1	423.34	50.0	—	1	435.57	38.6	—	1
041_B_half	40	404.09	222.5	—	1	411.73	187.6	—	1	423.34	44.8	—	1	435.57	47.5	—	1
041_B_one	40	404.09	158.9	—	1	411.73	78.2	—	1	423.34	42.3	—	1	435.57	37.7	—	1
041_B_two	40	404.09	146.9	—	1	411.73	68.0	—	1	423.34	55.8	—	1	435.57	23.2	—	1
045_B_p_two	44	665.54	156.3	—	2	662.40	17.6	—	1	666.98	24.4	—	1	672.48	3.2	—	1
045_B_half	44	665.54	108.0	—	2	662.40	60.4	—	1	666.98	24.0	—	1	672.48	3.7	—	1
045_B_one	44	665.54	112.7	—	2	662.40	6.6	—	1	666.98	6.0	—	1	672.48	2.3	—	1
045_B_two	44	665.54	138.8	—	2	662.40	10.0	—	1	666.98	8.0	—	1	672.48	2.4	—	1
048_B_p_two	47	37806.13	t.lim.	2.40	6	38436.23	t.lim.	2.02	7	38940.60	1170.7	—	6	40533.68	907.7	—	6
048_B_half	47	37907.98	t.lim.	2.66	7	38076.73	2976.8	—	7	38940.60	598.2	—	6	40533.68	653.6	—	6
048_B_one	47	37331.33	t.lim.	0.02	7	38076.73	3444.3	—	7	38940.60	790.8	—	6	40533.68	773.8	—	6
048_B_two	47	37331.33	t.lim.	1.38	7	38076.73	2680.0	—	7	38940.60	695.4	—	6	40533.68	737.3	—	6
051_B_p_two	50	462.15	101.1	—	1	466.23	42.3	—	1	480.04	62.0	—	1	494.01	54.4	—	1
051_B_half	50	463.26	116.4	—	1	467.34	87.2	—	1	480.04	97.2	—	1	494.01	60.3	—	1
051_B_one	50	463.26	103.9	—	1	467.34	62.6	—	1	480.04	97.1	—	1	494.01	41.8	—	1
051_B_two	50	463.26	106.7	—	1	467.34	41.4	—	1	480.04	73.7	—	1	494.01	30.4	—	1

Table 2.35: Detailed results for Table 2.8: GLS-Multi set, $\alpha = 0.4$, $71 \leq n \leq 100$ ($n = |PD|$)).

		$k=3$				$k=4$				$k=5$				$k=6$			
instance	|PD|	UB	sec	gap	it	UB	sec	gap	it	UB	sec	gap	it	UB	sec	gap	it
072_B_p_two	71	225.85	2695.2	—	3	235.47	181.6	—	1	246.87	63.7	—	1	259.16	59.5	—	1
072_B_half	71	225.85	915.2	—	3	235.47	49.3	—	1	246.87	65.3	—	1	259.16	50.1	—	1
072_B_one	71	225.85	544.9	—	3	235.47	73.1	—	1	246.87	81.0	—	1	259.16	36.4	—	1
072_B_two	71	225.85	785.8	—	3	235.47	94.9	—	1	246.87	63.7	—	1	259.16	40.4	—	1
076_B_p_two	75	575.37	2404.3	—	1	581.18	1443.1	—	1	590.74	t.lim.	0.20	1	600.54	534.0	—	1
076_B_half	75	575.88	1964.4	—	1	581.33	1358.8	—	1	590.74	2147.2	—	1	600.54	654.1	—	1
076_B_one	75	575.88	1841.4	—	1	581.33	1526.4	—	1	590.74	2312.7	—	1	600.54	865.3	—	1
076_B_two	75	575.88	1533.2	—	1	581.33	1050.3	—	1	590.74	1296.5	—	1	600.54	733.5	—	1
101_B_p_two_a	100	667.50	740.4	—	1	674.58	469.7	—	1	682.35	199.2	—	1	693.94	142.3	—	1
101_B_half_a	100	667.58	581.4	—	1	674.67	934.8	—	1	682.44	254.9	—	1	693.94	218.8	—	1
101_B_one_a	100	667.58	767.2	—	1	674.67	485.2	—	1	682.44	248.3	—	1	693.94	76.8	—	1
101_B_two_a	100	667.58	423.7	—	1	674.67	1067.5	—	1	682.44	549.2	—	1	693.94	100.0	—	1
101_B_p_two_b	100	555.73	2971.1	—	1	573.93	2288.5	—	1	594.30	1495.5	—	1	614.98	676.4	—	1
101_B_half_b	100	560.93	t.lim.	0.89	1	575.14	2621.5	—	1	594.61	1795.5	—	1	614.98	513.0	—	1
101_B_one_b	100	556.38	2850.5	—	1	575.14	2817.1	—	1	594.61	1680.7	—	1	614.98	504.6	—	1
101_B_two_b	100	556.38	1930.9	—	1	578.07	t.lim.	0.56	1	594.61	1409.4	—	1	614.98	696.6	—	1

2.5 Bibliography

M. Battarra, J.-F. Cordeau, and M. Iori. Pickup and delivery problems for goods transportation. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, pages 161–192. SIAM, 2nd edition, 2014.

J.F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15:1–31, 2007.

B.P. Bruck and A.G. dos Santos. Hybrid approach for the multiple vehicle routing problem with deliveries and selective pickups. In *12th International Conf. on Hybrid Intelligent Systems, (HIS 2012)*, pages 265–270, 2012.

B.P. Bruck, A.G. dos Santos, and J.E.C. Arroyo. Metaheuristics for the single vehicle routing problem with deliveries and selective pickups. In *12th International Conference on Intelligent Systems Design and Applications (ISDA 2012)*, pages 723–728, 2012.

D. Chemla, F. Meunier, and R. Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10:120 – 146, 2013.

G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 2006.

I.M. Coelho, P.L.A. Munhoz, M.N. Haddad, M.J.F. Souza, and L.S. Ochi. A hybrid heuristic based on general variable neighborhood search for the single vehicle routing problem with deliveries and selective pickups. *Electronic Notes in Discrete Mathematics*, 39:99–106, 2012.

J.F. Cordeau, M. Dell'Amico, S. Falavigna, and M. Iori. A rolling horizon algorithm for auto-carrier transportation. *Transportation Research Part B*, 76:68 – 80, 2015.

J.-F. Côté, M. Dell'Amico, and M. Iori. Combinatorial benders' cuts for the strip packing problem. *Operations Research*, 62:643–661, 2014.

V. Daniel, R. Guide Jr., and L.N. Van Wassenhove. The evolution of closed-loop supply chain research. *Operations Research*, 57:10–18, 2009.

K. Doerner and J.-J. Salazar-González. Pickup and delivery routing problems for people transportation. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, pages 193–212. SIAM, 2nd edition, 2014.

D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits: An overview. *Transportation Science*, 39:188–205, 2001.

B.L. Golden and A.A. Assad. Or forum – perspectives on vehicle routing: Exciting new developments. *Operations Research*, 34(5):803–810, 1986.

B.L. Golden, A.A. Assad, and E.A. Wasil. Routing vehicles in the real world: Applications in the solid waste, beverage, food, dairy, and newspaper industries. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 245–286. SIAM, Philadelphia, PA, 2002.

I. Gribkovskaia and G. Laporte. One-to-many-to-one single vehicle pickup and delivery problems. In B. Golden, S. Raghavan, E. Wasil, R. Sharda, and S. Voss, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 359–377. Springer, Berlin, 2008.

I. Gribkovskaia, Ø. Halskau, G. Laporte, and M. Vlcek. General solutions to the single vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, 180(2):568–584, 2007.

I. Gribkovskaia, G. Laporte, and A. Shyshou. The single vehicle routing problem with deliveries and selective pickups. *Computers & Operations Research*, 35:2908–2924, 2008.

G. Gutiérrez-Jarpa, V. Marianov, and C. Obreque. A single vehicle routing problem with fixed delivery and optional collections. *IIE Transactions*, 41:1067–1079, 2009.

H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145:126–139, 2004.

S. Irnich, C. Laganà, D.and Schlebusch, and F. Vocaturo. Two-phase branch-and-cut for the mixed capacitated general routing problem. *European Journal of Operational Research*, 243:17 – 29, 2015.

K. Kaparis and A. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical Programming*, 124:69–91, 2010.

G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2–3):193–207, 1990.

C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations and traveling salesman problems. *Journal of the ACM*, 7:326–329, 1960.

G. Nagy and S. Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162: 126–141, 2005.

M. Nowak, Ö. Ergun, and C.C. White. Pickup and delivery with split loads. *Transportation Science*, 42:32–43, 2008.

U. Pferschy and R. Stanek. Using pure integer solutions to solve the traveling salesman problem. Technical report, Department of Statistics and Operations Research, University of Graz, 2014. Available as: Optimization Online 2014-02-4258.

J. Privé, J. Renaud, F. Boctor, and G. Laporte. Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of Operational Research Society*, 57:1045–1052, 2006.

J.-J. Salazar-González and B. Santos-Hernández. The split-demand one-commodity pickup-and-delivery travelling salesman problem. *Transportation Research Part B*, 75:58 – 73, 2015.

A. Subramanian, E. Uchoa, A.A. Pessoa, and L.S. Ochi. Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. *Operations Research Letters*, 39:338–341, 2011.

H. Süral and J.H. Bookbinder. The single-vehicle routing problem with unrestricted backhauls. *Networks*, 41:127–136, 2003.

L.A. Wolsey. *Integer Programming*, volume 52 of *Wiley Series in Discrete Mathematics and Optimization*. Wiley, New York, NY, 1998.

Chapter 3

Solving the static bike sharing rebalancing problem without temporary operations

In this chapter we concentrate on a vehicle routing problem found in the context of bike sharing systems, which is a fast growing market with hundreds of initiatives spread all over the world. These systems are growing on importance due to the fact that they provide an interesting tool for governments to support sustainable mobility, reduce urban traffic and pollution. In particular, we approach the *static bike sharing rebalancing problem without temporary operations* (SBRPW), in which a single vehicle based on a depot must perform a series of pickups and deliveries to bring stations to a balanced state. Although multiple visits and split demands are allowed at unbalanced stations, temporary deliveries and temporary pickups are strictly forbidden. To our knowledge this is the first work to approach the SBRPW, then we formally introduce the problem, present a few interesting theoretical results and propose two exact algorithms. Our algorithms are tested on a set of benchmark instances derived from the related literature and results show that our approaches are able to solve to optimality the majority of instances in a reasonable amount of time.

3.1 Introduction

In the *static bike rebalancing problem without temporary operations* (SBRPW) we are given a directed graph $G = (V, A)$, where $V = I \cup \{0\}$ is the set of vertices, $I = \{1, ..., n\}$ is the set of n bike stations, and 0 is the depot, and $A = \{(i, j) : i, j \in V, i \neq j\}$. Each station $i \in I$ has a certain demand d_i of bikes, and is said to be in *excess* if $d_i > 0$, in *default* if $d_i < 0$, or *initially balanced* if $d_i = 0$. The depot is always assumed to have no bikes, i.e, $d_0 = 0$. Each arc in $(i, j) \in A$ is associated with a travelling cost c_{ij}, possibly asymmetric but satisfying

the triangle inequalities. The system is considered to be balanced, i.e., $\sum_{i \in I} d_i = 0$.

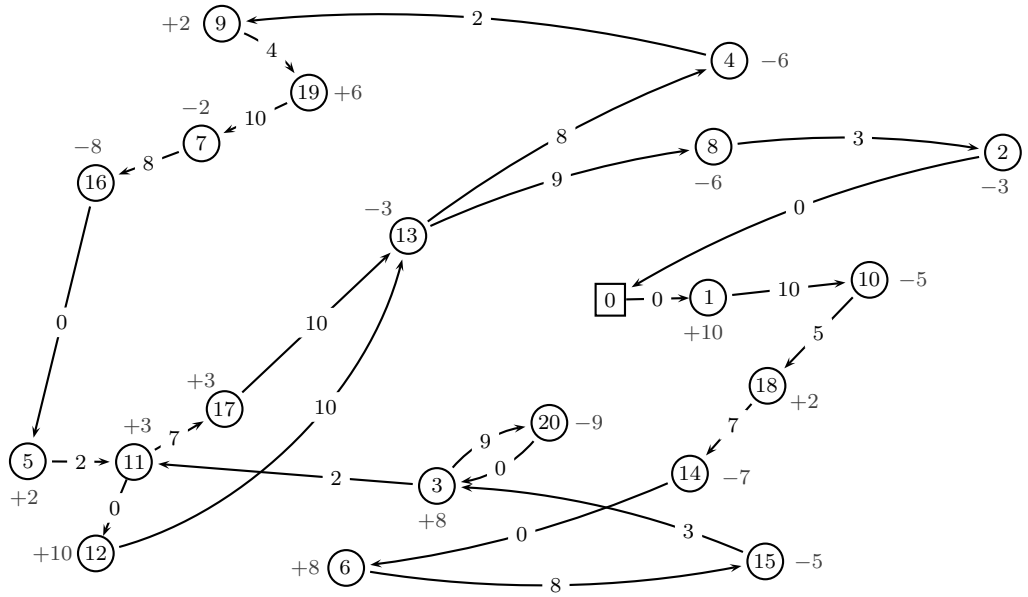
The SBRPW aims at finding a minimum cost route in which a capacitated vehicle performs a series of pickups and deliveries to satisfy the demand of each station, while ensuring that the following constraints are not violated: (i) the route starts and ends at the depot with the vehicle empty; (ii) the vehicle capacity is not exceeded at any point of the route; (iii) the demand of each station is completely fulfilled; (iv) and no temporary operations (temporary pickups or deliveries of bikes) are performed. Furthermore, we assume that initially balanced stations need not to be visited. However, multiple visits and split of demand is allowed at unbalanced stations.

To our knowledge this is the first work to directly approach the SBRPW. However, its closest variant, called the *static bicycle rebalancing problem* (SBRP), where temporary operations are allowed, was studied by Chemla et al. (2013) and Erdoğan et al. (2015). Details about their proposed approaches are provided in the following section. Our study is motivated by the fact that temporary operations require additional manual work and service time, and are frequently disliked by the bike sharing system operators. That is indeed what was found in the real world application studied by Dell'Amico et al. (2014). The problem they address, called the *bike rebalancing problem* (BRP), is another close variant of the SBRPW, where multiple vehicles are used but multiple visits to stations are not allowed.

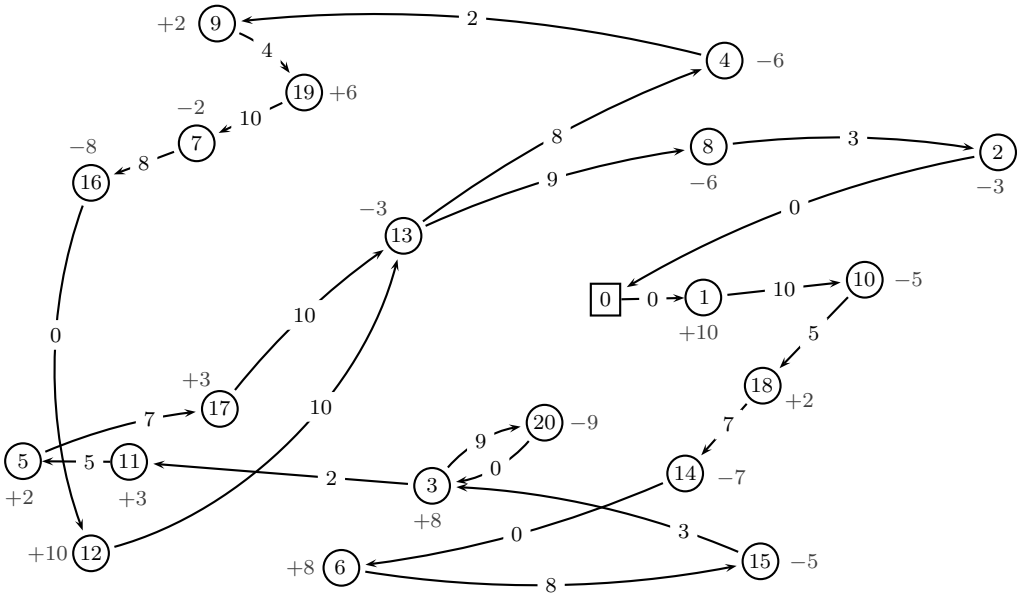
By forbidding temporary pickups and deliveries, the complexity of operations at stations is reduced, but routing costs might be higher than in the SBRP. As an illustrative example, consider Figure 3.1-(a), which shows an optimal solution for the SBRP on a benchmark instance. Each arc is traversed once and the flow of commodity passing through the arc is shown nearby. At the first visit to station 11, instead of performing a pickup demand, the vehicle temporally deliveries 2 bikes. Then, at the second visit the demand of station 11 is performed and the 2 units delivered in the first visit are also collected. In Figure 3.1-(b) we present the optimal solution for the same instance, but for the SBRPW. Notice that forbidding temporary operations resulted in an increase on routing costs from 6012 to 6025, but it also substantially reduced the complexity of operations at station 11.

It is worth mentioning that the SBRPW is not limited to bike sharing systems and can be found in other contexts, such as courier service transportation and the repositioning of inventory between retail stores. In these cases, however, it might be more practical to call it the *single vehicle one-commodity capacitated pickup and delivery problem without temporary operations*.

In this chapter we propose two exact algorithms for the SBRPW. The first one is based on an aggregated formulation and aims at solving the SBRPW by duplicating as few vertices as possible so as to avoid temporary operations, whereas the second is built upon an arc structure which enables us to only use binary variables and infeasible path constraints to forbid temporary operations. More details on these approaches are provided in Sections 3.4



(a) Optimal solution with temporary operations with value 6012 for instance n20C with $Q = 10$



(b) Optimal solution without temporary operations with value 6025 for instance n20C with $Q = 10$

Figure 3.1: Example of optimal solution for the SBRP and SBRPW on a real benchmark instance.

and 3.5. Furthermore, we also present a few interesting theoretical results on the SBRPW and propose an efficient feasibility check procedure that is used by both algorithms.

The remainder of this chapter is organized as follows. In Section 3.2 the related literature is reviewed. In Section 3.3, the complexity of the problem is discussed and theoretical results are presented. Then, in Section 3.4 and 3.5 our two exact algorithms are presented. In Section 3.6 computational results are shown and in Section 3.7 some conclusions are drawn.

3.2 Literature review

Nowadays, there is an ever growing trend in self-service bike sharing systems. From the first bike sharing system implemented in Amsterdam in 1965 (see, e.g., DeMaio 2009), their numbers has greatly increased reaching up to about 400 systems by 2011, only in Europe (see, e.g., OBIS project 2011). These systems are an important tool in supporting sustainable mobility and the reduction of urban traffic and pollution.

According to the classification scheme proposed by Berbeglia et al. (2007), the SBRPW is a *many-to-many pickup and delivery problem* (M-M-PDP), in which commodities may have multiple origins and destinations and any location might serve as origin or destination.

In the literature, there are a few related M-M-PDPs. For instance, the *one-commodity pickup and delivery travelling salesman problem* (1-PDTSP), where a capacitated vehicle must perform a series of pickups and deliveries of a single commodity. The main differences with respect to the SBRPW, are that in the 1-PDTSP all customers must be visited exactly once and the depot serves as a customer, thus split of demand is not allowed and the vehicle is not required to departure empty from the depot. The 1-PDTSP was introduced by Hernández-Pérez and Salazar-González (2004a). The authors proposed a mathematical formulation for both the symmetric and the asymmetric versions of the problem, and a branch-and-cut algorithm. This approach was later improved by the same authors in Hernández-Pérez and Salazar-González (2007), where they proposed another branch-and-cut algorithm and presented valid inequalities to strengthen their formulation. For a few examples of heuristic approaches to this problem we refer the interested reader to the works of Hernández-Pérez and Salazar-González (2004b), Hernández-Pérez et al. (2009), Wang et al. (2006) and Mladenović et al. (2012).

In Erdoğan et al. (2014), the authors introduce a variant of the 1-PDTSP called the *static bicycle relocation problem with demand intervals* (SBRP-DI), where stations might be visited multiple times and the number of bikes at stations after the rebalancing may lie within a given interval instead of a fixed target value. The authors propose two exact methods to solve the problem and present valid inequalities adapted from related problems.

Another related problem is the BRP, where a fleet of vehicles is used to serve a set of customers requests, while minimizing the routing costs. In the BRP, stations must be visited

exactly once and the vehicles may leave the depot with some initial load. This problem has been recently studied by Dell'Amico et al. (2014). The authors presented four mathematical formulations and branch-and-cut algorithms to solve the problem. In Dell'Amico et al. (2016), the same authors developed a destroy and repair heuristic, improving the best known solutions for several instances of the literature.

A dynamic variant of the BRP was studied by Contardo et al. (2012), in which the demand of stations is not static and the fleet of vehicles is heterogeneous. The authors presented a mathematical formulation to formally introduce the problem and proposed a solution approach based on an arc flow formulation on a time horizon, solved using Dantzig-Wolfe and Benders decompositions.

Recently, Salazar-González and Santos-Hernández (2015) proposed the *split-demand one-commodity pickup and delivery travelling salesman problem* (SD1PDTSP), which is a generalization of the 1-PDTSP. In the SD1PDTSP, stations and the depot might be visited multiple times by a single vehicle, the depot is considered as a customer having demand and capacity, and it is not imposed that the vehicle should either depart from or arrive at the depot empty. In addition, notice that a maximum number of visits is imposed to vertices, and split of demand and temporary operations are allowed. To solve the problem, the authors proposed a branch-and-cut (B&C) algorithm, further strengthened by the use of Benders decomposition and a few valid inequalities.

Chemla et al. (2013) studied the *single vehicle one-commodity capacitated pickup and delivery problem* (SVOCPDP), in which the vehicle departs empty from the depot to perform a series of pickups and deliveries in order to bring each station from an initial to a target state. In the SVOCPDP, split of demand and temporary operations are allowed, however there is no limit to the number of times a station might be visited and it is not mandatory to visit stations with null demand. The authors proposed a complete formulation to define the problem, presented two relaxations to provide lower bounds and proposed a *tabu search* algorithm.

The SVOCPDP has also been studied in a later work by Erdoğan et al. (2015). The authors referred to the problem as the SBRP and proposed an exact algorithm to solve it. In addition, they also studied the non-preemptive version of the SBRP, where a bike can be loaded on the vehicle and unloaded at a station at most once, in contrast with the SBRP, where bikes can be temporarily loaded and unloaded any number of times. In our case however, temporary operations are strictly forbidden. In practice, the situation depicted in Figure 3.1-(a) is feasible for both the preemptive and the non-preemptive versions studied by Erdoğan et al. (2015), but it is infeasible for the SBRPW.

3.3 Preliminary results

It is not difficult to see that the SBRPW is an NP-hard problem, as the *travelling salesman problem* (TSP) is a special case of it. In order to prove this statement let us consider a TSP instance with n vertices from $1, 2, \ldots, n$ and then let us show that it can be transformed into an SBRPW instance as follows: assign a pickup demand $d_1 = n - 1$, a delivery demand $d_i = -1$ for each $i \in \{2, \ldots, n\}$ and assume the vehicle capacity as $Q = n - 1$. By placing a depot at the same position as station 1, the optimal solution of the SBRPW corresponds to the solution of the TSP.

Because unbalanced stations can be visited an arbitrarily number of times, the size of the route may grow exponentially with respect to the number of stations. Chemla et al. (2013) proved that, when the triangle inequalities hold, any arc connecting two stations with both positive or negative demand values is traversed at most once. In a recent paper, Erdoğan et al. (2015) showed that arcs $(i, j) \in A$, such that $d_i > 0$ and $d_j < 0$, or $d_i < 0$ and $d_j > 0$, are traversed no more than $min(|d_i|, |d_j|)$ times. In this sense, let u_{ij} be an upper bound on the number of traversals on arc $(i, j) \in A$, defined as

$$u_{ij} = \begin{cases} min(|d_i|, |d_j|) & \text{if } (d_i > 0 \text{ and } d_j < 0) \text{ or } (d_i < 0 \text{ and } d_j > 0), \\ 1 & \text{otherwise.} \end{cases} \tag{3.1}$$

Furthermore, notice that each station $i \in I$ might be visited at most $\beta_i = |d_i|$ times, which accounts for the situation in which at each visit one unit of demand is performed.

3.3.1 Aggregated formulation

We first introduce what is probably the best lower bound for the SBRPW, based on an aggregated formulation defined as follows.

For any set $S \in V$, let $\delta^+(S) = \{(i, j) \in A : i \in S, j \in V \setminus S\}$, $\delta^-(S) = \{(i, j) \in A : i \in V \setminus S, j \in S\}$ and $(\bar{S} : S) = \{(i, j) \in A : i \in \bar{S}, j \in S\}$. In addition, because we consider the triangle inequalities to hold, from now on assume that initially balanced stations are effectively removed from I. Then, consider x_{ij} as an integer variable specifying the number of times arc (i, j) is traversed and let us present the following *aggregated formulation* (AF).

$$\text{(AF)} \qquad \min \; z_{\text{AF}} = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.2}$$

subject to

$$x(\delta^+(0)) = 1 \tag{3.3}$$

$$x(\delta^+(i)) \geq 1 \qquad \forall \; i \in I \tag{3.4}$$

$$x(\delta^-(i)) - x(\delta^+(i)) = 0 \qquad \forall \; j \in V \tag{3.5}$$

$$x(\bar{S} : S) \geq max\left(\left\lceil \frac{|\sum_{i \in S} d_i|}{Q} \right\rceil, 1\right) \qquad \forall\ S \subseteq I, \bar{S} = I \setminus S \qquad (3.6)$$

$$x_{ij} \in \{0, 1, .., u_{ij}\} \qquad \forall\ (i,j) \in A \qquad (3.7)$$

The objective function (3.2) minimizes the total distance travelled. Constraints (3.3) and (3.4) define degree constraints for the depot and stations, while (3.5) specify flow conservation constraints for the vehicle. Constraints (3.6) define the maximum capacity and (3.7) define the integrality of the x variables.

The main reason that prevents this formulation from exactly solving the SBRPW is the fact that it does not consider the evolution on the number of bikes at each visit to a station. Consequently, it might produce solutions with temporary operations, and others where there are no temporary operations but the associated bike displacements are still not feasible. An example of the second case is depicted in Figure 3.2, where $I = \{1, 2, 3, 4, 5\}$ and each arc shown is traversed only once. Although this solution is feasible for formulation AF, it is clearly infeasible for the SBRPW. Furthermore, notice that both shortcomings may only exist in solutions where there is at least one vertex being visited multiple times.

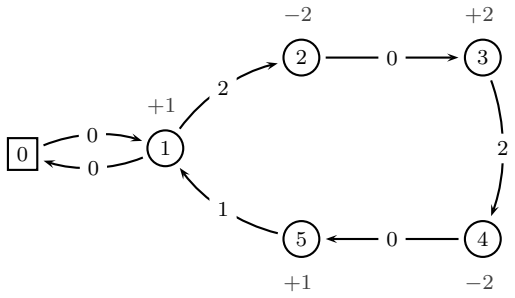


Figure 3.2: Example of infeasible aggregated solution without temporary operations.

Consider the following definitions.

Definition 3 Given a directed graph $G = (V, A)$, the *extended network* is the graph obtained by duplicating each station $i \in I$ into $|d_i|$ duplicates.

Definition 4 An *aggregated solution* is that obtained by solving the SBRPW under a graph where at least one vertex can be visited multiple times, whereas a *disaggregated solution* is one obtained by solving the SBRPW under the extended network, where vertices are visited at most once.

A straightforward approach to define a complete formulation can be achieved by solving the SBRPW under the extended network. By doing so we guarantee that no vertices are visited more than once, thus eliminating the aforementioned shortcomings that may render

solutions infeasible. However, such a formulation becomes easily intractable due to the high number of duplicate vertices, even for small size instances.

3.3.2 Feasibility check

First consider the following definition.

DEFINITION 5 An aggregated solution $(\bar{z}, \bar{x})$ of value $\bar{z}$ is said to be *induced* by a disaggregated solution $(\bar{z}^d, \bar{x}^d)$ when $\bar{x}(\delta^+(i)) = \bar{x}^d(\delta^+(i))$ and $\bar{z} = \bar{z}^d$.

Another difficult regarding the SBRPW refers to verifying the feasibility of solutions. Notice that, given a sequence of vertices, there is a polynomial algorithm to determine whether or not this sequence is induced by a feasible solution of the SBRPW. However, if we are given an aggregated solution, determining whether or not this solution is induced by a feasible disaggregated solution of the same value is a strongly NP-complete decision problem. These properties are formalized in the following along with their respective proofs.

PROPERTY 9 *Let $r = r_1, r_2, \ldots, r_k$ be a sequence of vertices defining a route that starts and ends at the depot. There is a polynomial algorithm to decide whether or not r is induced by a feasible solution for the SBRPW.*

Proof Chemla et al. (2013) proved that, for the SBRP there is a polynomial algorithm to find the best bike displacements that are compatible with route r, and to verify whether or not r is induced by a feasible solution. We now extend this result to the SBRPW by introducing a modified version of their algorithm. Let us define a support graph $G' = (V', A')$, where V' is composed by as many duplicates of each station $i \in I$ as there are occurrences of i in route r, and two additional vertices s and t. Then, the set A' contains the following arcs:

1. one arc between s and the first occurrence of each station $i \in I$ with capacity $max(0, d_i)$,
2. one arc (r_j, r_{j+1}) for each $j = 1, \ldots, k-2$, with capacity Q,
3. one arc (i_j, i_{j+1}) with capacity $|d_i|$ between the j-th occurrence of each station i and its $(j+1)$-th occurrence, if any,
4. one arc between the last occurrence of each station i and t, with capacity $max(0, -d_i)$.

As an illustrative example, consider a route defined by the sequence $0 \to 1 \to 2 \to 3 \to 2 \to 1 \to 2 \to 3 \to 0$. Figure 3.3 shows the resulting support graph G' for this route. Arcs created in step 1. represent the initial inventory at each station, while arcs produced in step 2. represent the load on the vehicle at each arc of the route. The remaining amount of bikes at stations after each visit is represented by arcs created in step 3. and the final inventory at stations is represented by arcs produced in step 4. Then, evaluating the maximum s-t flow

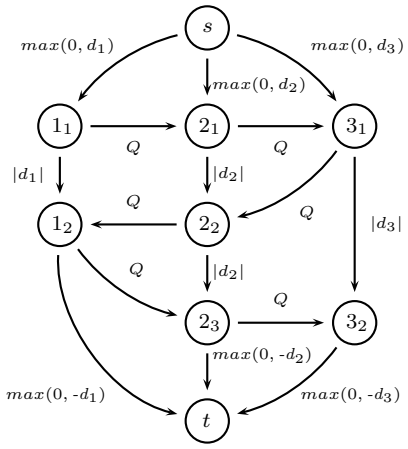


Figure 3.3: Example of the graph built by the procedure of Property 9

in G' yields the best bike displacements that are compatible with the sequence of vertices in route r, do not violate the vehicle capacity, and do not contain temporary operations.

Let $\bar{f}(i,j)$ be the resulting flow passing through arc $(i,j) \in A'$. To verify whether or not r is induced by a feasible solution for the SBRPW, it suffices to check that $\bar{f}(s,i_1) = max(0,d_i)$ for each station $i \in I$. Furthermore, notice that the bike displacements associated with r can be retrieved by the values $\bar{f}(r_j, r_{j+1})$ for each $j = 1,\ldots,k-2$. □

3.3.3 Complexity of building a disaggregated solution from an aggregated one

Consider the following property,

PROPERTY 10 *Given an aggregated solution $(\bar{z},\bar{x})$ of value $\bar{z}$, determining if it is induced by a feasible SBRPW disaggregated solution of the same value is an NP-Complete problem.*

Proof This property has been proved for the SBRP by Chemla et al. (2013) and their result is directly applicable to our problem. However, we propose here a slightly different version because it serves as a basis for the proof of Property 11.

The *partition problem* is defined as follows: given n items of weight w_j and $\sum_{j=1}^{n} w_j = 2b$, we must decide whether or not there exists a partition on the items into two sets (S_1, S_2), such that $\sum_{i\in S_1} w_i = \sum_{i\in S_2} w_i = b$. This problem is known to be NP-Complete (see Garey and Johnson 1979) and in the following we prove that it can be reduced into the SBRPW.

Consider the aggregated solution of value $\bar{z} = n$ depicted in Figure 3.4. Let $\{1,\ldots,n\}$ be a set of n stations in excess with demand $d_i = w_i$, consider another $2n+1$ initially balanced stations, represented by vertices in dashed lines, and let $k = 2n+1$ be a station in default with demand $d = -2b$. Define the vehicle capacity as $Q = b$. Horizontal and diagonal arcs

have travel cost 0, while vertical ones have cost 1/2. Arcs not shown are considered to have a very high cost and are not used. In addition, notice that values nearby or on arcs represent the total load transported by the vehicle. Consider that the triangle inequalities do not hold and, consequently, stations with null demand may be visited.

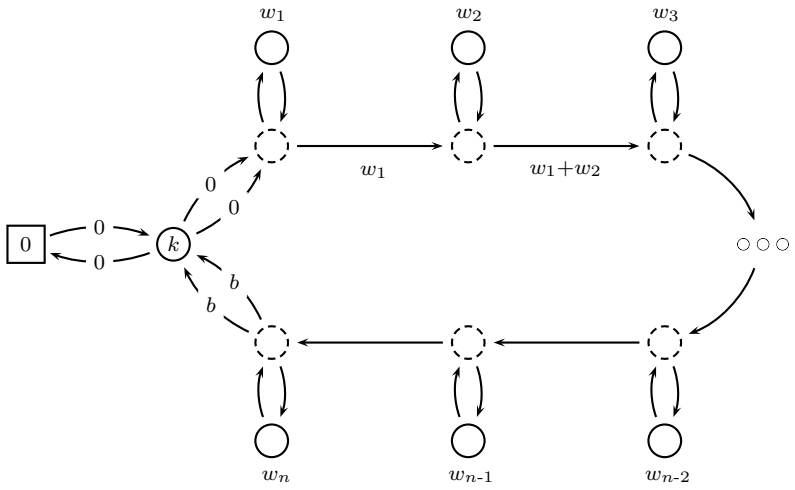


Figure 3.4: Transformation of the *partition problem* into the SBRPW. Horizontal and diagonal arcs have cost 0 and vertical arcs cost 1/2.

In case there is a disaggregate solution with value n, the vehicle uses each vertical arc exactly once, without performing any split of demand. Therefore, the answer to the partition problem is *"yes"*. If instead, the disaggregated solution has cost greater than n, then at least one vertical arc is used more than once and thus the vehicle performs split demands. Therefore, it is not possible to partition the set of items into (S_1, S_2) such that $\sum_{i \in S_1} w_i = \sum_{i \in S_2} w_i = b$, and the answer to the partition problem is *"no"*. □

PROPERTY 11 *Given an aggregated solution of value $\bar{z}$, determining whether or not it is induced by a feasible SBRPW disaggregated solution of the same value is a strongly NP-Complete problem.*

Proof The *3-partition problem* is defined as follows: given $n = 3m$ items of weight w_i, where $\sum_{i=1}^{n} w_i = mb$, we must decide whether or not there is a partition of the items into m triplets, such that each triplet S has $\sum_{i \in S} w_i = b$. In the following we prove that determining whether or not an aggregated solution is induced by a feasible disaggregated solution of the same value, is at least as hard as solving the 3-partition problem, which is known to be *strongly* NP-Complete (see Garey and Johnson 1979).

Consider the aggregated solution of value $\bar{z} = n$ shown in Figure 3.5. Let $\{1, \ldots, n\}$ be a set of n stations in excess with demand $d_i = w_i + M$, where M is a very large value, at least one order of magnitude greater than the greatest value in w. In addition, consider another $2n + 1$ initially balanced stations, represented by vertices in dashed lines, and let $k = 2n + 1$

be a station in default with demand $d = -\sum_{i=1}^{n} d_i$. There are m trips out and back from station k and notice that,

$$\sum_{i=1}^{n} d_i = nM + mb = 3mM + mb = m(3M + b). \tag{3.8}$$

Then, define the vehicle capacity as $Q = 3M + b$ and consider the same assumptions on the cost of arcs as specified in Property 10.

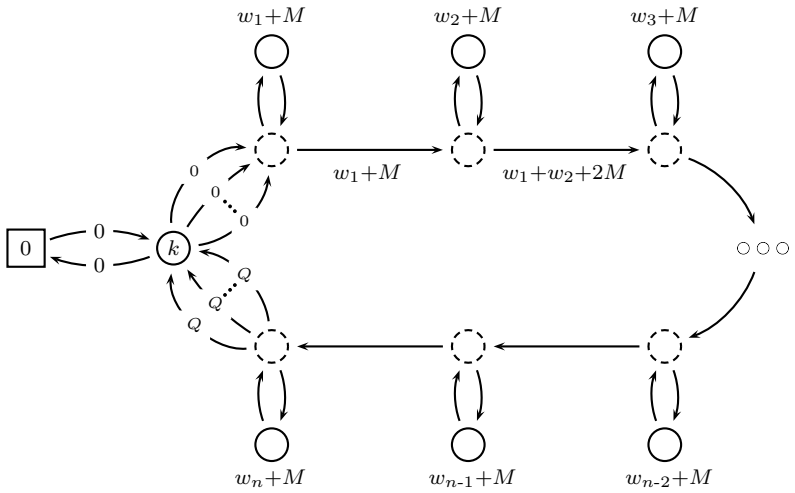


Figure 3.5: Transformation of the *3-partition* problem into the SBRPW. Horizontal and diagonal arcs have cost 0 and vertical arcs cost 1/2.

In case there exists a disaggregated solution having value $\bar{z} = n$, each vertical arc is traversed exactly once and at each trip out and back from station k, the vehicle visits exactly 3 stations and picks up $3M + b$ units of the commodity, which are latter delivered to k. Therefore, the answer to the 3-partition problem is *"yes"*. On the other hand, if no such solution exists, a split demand was performed and the vehicle visited more than 3 stations at least once in a tour out and back from k. Then, the answer to the 3-partition problem is *"no"*. □

3.3.4 Worst case analysis

In the *vehicle routing problem* (VRP) we are given a fleet of vehicles that must visit a series of customers exactly once to perform a delivery demand, whereas in its generalization called the *split delivery vehicle routing problem* (SVRP), customer demands may be split and served through multiple visits. Let $z(VRP)$ and $z(SDVRP)$ be the optimal solution values for the VRP and the SDVRP, respectively. Archetti et al. Archetti et al. (2006) showed that $z(VRP)/z(SDVRP) \leq 2$ and proved that this bound is tight. This shows that allowing splits

may halve the VRP solution cost.

Nowak et al. Nowak et al. (2008) studied the *pickup and delivery problem with split loads* (PDPSL) in which a single vehicle picks up load from specific origins and delivery to their destination. This problem generalizes the *pickup and delivery problem* (PDP) by allowing demand to be split. The authors then propose a conjecture stating that $z(PDP)/z(PDPSL) \leq 2$.

We could not find a similar conjecture for the SBRPW and the proof in Archetti et al. Archetti et al. (2006) does not extend to the SBRPW. However, we did discover an interesting property relating the SBRP with its particular case where split of demand is not allowed, called the *static bike rebalancing problem without split demand* (SBRPWS). This property is shown in the following.

PROPERTY 12 *Split of demand in the SBRP may lead to an arbitrarily large gain in the solution value.*

Proof Consider the SBRP instance depicted in Figure 3.6. There is a single vehicle with capacity Q and $3Q + 2$ vertices with demand values shown nearby. These vertices include a depot 0, a pickup station p, and Q triplets. Each arc shown in the figure has cost 1, whereas all other arcs are assumed to have an arbitrarily large cost.

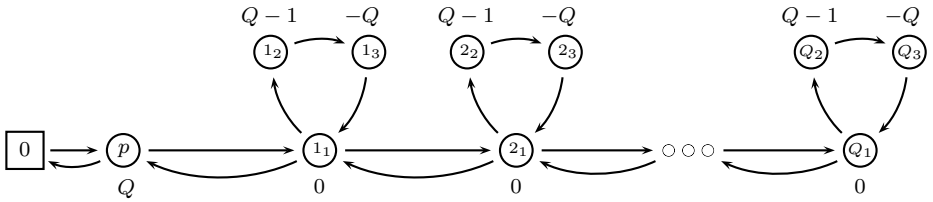


Figure 3.6: Instance of the SBRP with Q triplets.

The optimal solution for the SBRP in this instance is depicted in Figure 3.7. Each arc is used once and the value nearby specify the flow of commodity passing through it. At the first visit to station p the vehicle picks up Q units, then at 1_1 it performs a temporary delivery of $Q - 1$ units and in what follows the vehicle serves the other stations of this first triplet (i.e., 1_2 and 1_3). Returning to station 1_1, the vehicle undo the temporary delivery by picking up the $Q - 1$ units stored at this station. Next the vehicle proceeds to the following triplet and performs the same operations, including a dropoff of $Q - 2$ units at vertex 2_1. After serving the last triplet the vehicle returns empty to the depot. Then, the cost of such a solution is given by $z(SBRP) = \sum_{i=1}^{Q}(2 + 3) + 2 = 5Q + 2$.

Because in the SBRPWS split of demand is not allowed and, consequently, neither are temporary operations, the solution depicted in Figure 3.7 is infeasible for this problem. In fact, the optimal solution of the SBRPWS in this instance, shown in Figure 3.8, requires the vehicle to perform Q trips out and back from station p. At each of these trips, shown separately for the sake of simplicity, the vehicle performs the demands on one of the triplets.

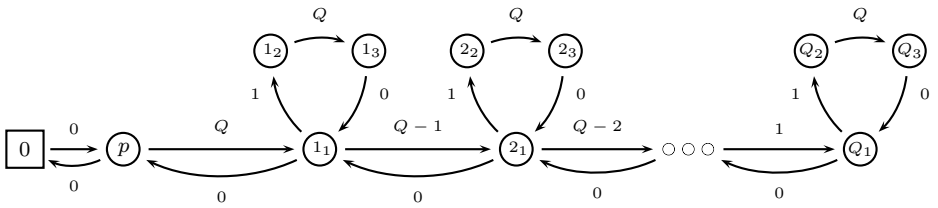


Figure 3.7: Optimal solution for the SBRP.

For instance, at the first trip, the vehicle departs from p loaded with 1 unit, goes to the first triplet to serve the demands of stations 1_2 and 1_3, and then returns empty to p. The cost of such a solution is then given by $z(SBRPWS) = \sum_{i=1}^{Q}(2i+3)+2 = Q(Q+1)+3Q+2$. Therefore we have that,

$$\frac{z(SBRPWS)}{z(SBRP)} = \frac{Q(Q+1)+3Q+2}{5Q+2} \xrightarrow[Q\to+\infty]{} +\infty. \tag{3.9}$$

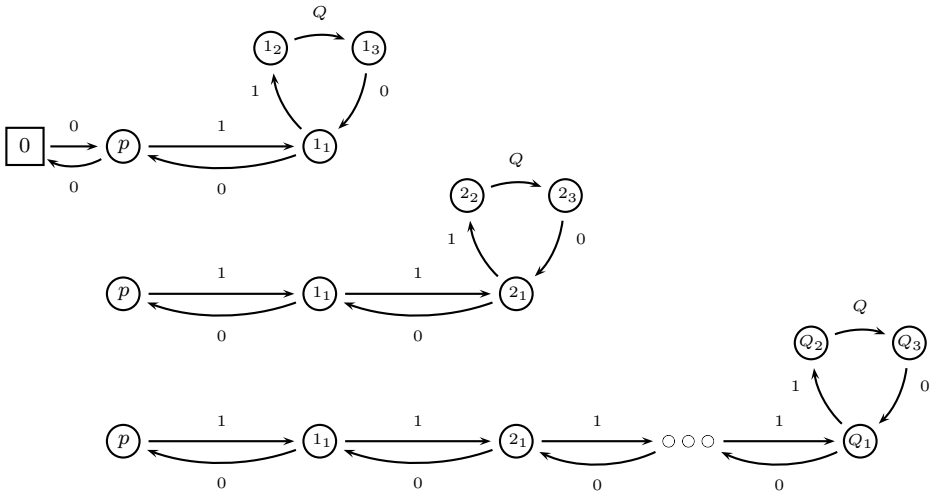


Figure 3.8: Optimal solution for the SBRPWS.

3.4 Binary arc formulation

In Erdoğan et al. (2015), the authors proposed an exact algorithm to solve the SBRP and further adapted it to the non-preemptive SBRP. In this section we present an extension of their algorithm which is capable of solving the SBRPW.

As mentioned in Section 3.1, the biggest issue preventing formulation AF from exactly solving the SBRPW is that it does not consider the evolution on the number of bikes at stations after each visit, given that stations may be visited multiple times. Because the x

variables defined for AF are integer, it is not possible to separate *infeasible path* constraints to forbid solutions with temporary operations. However, provided that the triangle inequalities hold, it is possible to transform each integral variable x_{ij} into a set of y_{ij}^k binary variables as in the following,

$$x_{ij} = \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ij}^k \qquad \forall\ (i,j) \in A \tag{3.10}$$

This can be intuitively seen as a representation of the value of x_{ij} as a binary number, where each variable y_{ij}^k represents the k-th bit (see also Vanderbeck 1999). As an illustrative example, consider Figure 3.9-(a), which shows an arc (i,j) that may be traversed at most 7 times, i.e., $u_{ij} = 7$. Applying transformation (3.10) to x_{ij} results in a set of 4 arcs as depicted in Figure 3.9-(b). Supposing the vehicle travels 5 times from vertex i to j, according to (3.10) we must select $y_{ij}^0 = y_{ij}^2 = 1$ so that $x_{ij} = 2^0 y_{ij}^0 + 2^1 y_{ij}^1 + 2^2 y_{ij}^2 + 2^3 y_{ij}^3 = 5$.

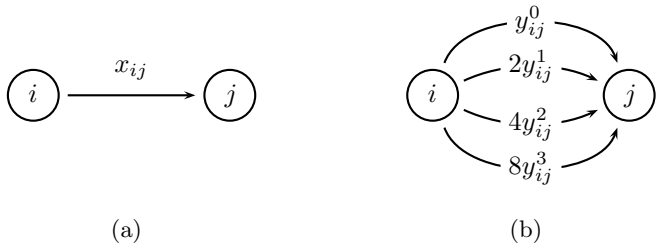


Figure 3.9: Example of disaggregation by the use of binary arcs.

Let $\mathcal{R}$ be the set of infeasible routes and consider $A(r) = \{(i,j,k) : (i,j,k) \in r\}$ as the set of arcs in route $r \in \mathcal{R}$. In addition, for the sake of simplicity let us define $\gamma_{ij} = \lfloor log_2(u_{ij}) \rfloor$. We are now ready to present the following *binary arc formulation* (BinArc).

$$\text{(BinArc)} \quad \min\ z_{\text{BinArc}} = \sum_{i \in V} \sum_{j \in V} \sum_{k=0}^{\gamma_{ij}} c_{ij} 2^k y_{ij}^k \tag{3.11}$$

subject to

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{i0}} 2^k y_{i0}^k = 1 \tag{3.12}$$

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k \geq 1 \qquad \forall\ j \in I \tag{3.13}$$

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k \leq u_{ij} \qquad \forall\ j \in I \tag{3.14}$$

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k - \sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ji}^k = 0 \qquad \forall\ j \in V \tag{3.15}$$

$$\sum_{i\in\bar{S}}\sum_{i\in S}\sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k \geq max\left(\left\lceil\frac{|\sum_{i\in S} d_i|}{Q}\right\rceil, 1\right) \qquad \forall\ S\subseteq I, \bar{S}=I\setminus S \qquad (3.16)$$

$$\sum_{(i,j,k)\in r} y_{ij}^k \leq |r|-1 \qquad \forall\ r\in\mathcal{R} \qquad (3.17)$$

$$y_{ij}^k \in \{0,1\} \qquad \forall\ (i,j)\in A, k=\{0,\ldots,\gamma_{ij}\} \qquad (3.18)$$

The objective function (3.11) minimizes the total distance travelled. Constraints (3.12)-(3.14) impose the degrees for the depot and the stations. Constraints (3.15) specify that each arrival at a vertex is followed by a departure, while (3.16) are capacity constraints for the vehicle. Infeasible routes with temporary operations are forbidden by constraints (3.17) and constraints (3.18) formally introduce the y variables. Notice that constraints (3.14) are not strictly necessary, but are used to strengthen the formulation, given that $\sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k$ might be greater than u_{ij}.

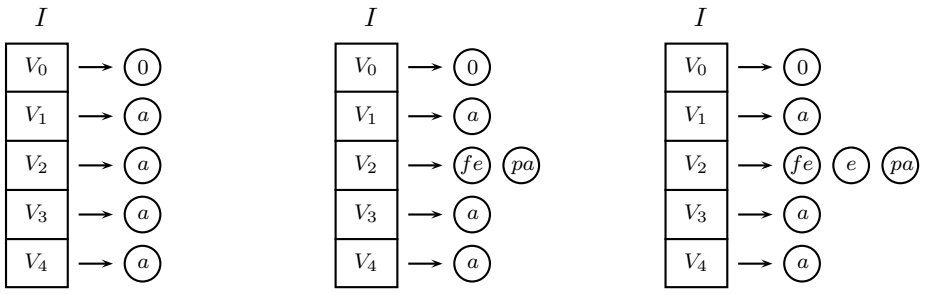
In order to separate the infeasible path constraints (3.17), Erdoğan et al. (2015) propose a depth-first algorithm that looks for Eulerian paths associated with a given aggregated solution. They check the feasibility of each Eulerian path found by solving a flow problem on the corresponding time-extended network, where each station is associated with as many duplicates as there are visits in it. Notice that, to prove that an aggregated solution is feasible, it suffices to find an Eulerian path for which the flow problem has a feasible solution. However, proving that a solution is infeasible involves finding and analyzing all possible paths that start and end at the depot, which might be very time consuming as the number of paths might be exponentially large. We opted to apply instead a feasibility check based on insights from the extended network, formally described in Section 3.5.1 below. Formulation (3.12)-(3.18) is thus solved by a B&C algorithm that separates cuts only at integer nodes of the branching tree.

3.5 The minimal extended network algorithm

In this section we introduce a formulation that is able to model any state of the network, from the simple aggregated case, where stations are represented by a single vertex, to the extended network scenario, where stations are represented by as many duplicates as necessary.

Let $G''=(V'',A'')$ be a complete directed graph, where $V''=\cup_{i\in I}V_i\cup\{0\}$ and $V_i=\{i_1,\ldots,i_{\bar{\beta}_i}\}$ is the set of vertices associated with station $i\in I$, such that $\bar{\beta}_i\leq\beta_i$. In addition, let I_j specify the station associated with vertex $j\in V''$. To represent the evolution of the network and the interactions between vertices, set V'' is partitioned into $V''=V^a\cup V^{pa}\cup V^{fe}\cup V^e\cup\{0\}$. Consider as an illustrative example an aggregated case with 4 stations, depicted in Figure 3.10-(a). Initially, the network contains only *aggregated vertices* (V^a). These vertices may be visited multiple times and, in case the associated station is either in excess or in default, a minimum of one visit is imposed. Now, suppose that in our example

vertex 2 is split, as shown in Figure 3.10-(b). The aggregated vertex is replaced by a *first extended vertex* (V^{fe}) and a *partially aggregated vertex* (V^{pa}). Vertices in V^{fe} are visited at most once, while those in V^{pa} may be visited multiple times. However, a partially aggregated vertex may only be visited when all other vertices associated with the same station are also visited. Furthermore, a second split of vertex 2, as shown in Figure 3.10-(c), results in an *extended vertex* (V^e), which may be visited at most once. Consider $k \in V^{fe}$ and $l \in V^e$, such that $I_k = I_l = i \in I$. The only difference between k and l is that l can only be visited when k is also. Additionally, in case $d_i \neq 0$, vertex k must be visited exactly once.



(a) Initial state of the network

(b) Network after splitting 2

(c) Network after splitting twice vertex 2

Figure 3.10: Example of consecutive splitting of vertex 2

We are now ready to introduce the following *general formulation* (GF), which is the stepping stone of our solution algorithm.

$$\text{(GF)} \quad \min \; z_{\text{GF}} = \sum_{i \in V''} \sum_{j \in V''} c_{ij} x_{ij} \tag{3.19}$$

subject to

$$x(\delta^+(0)) = 1 \tag{3.20}$$

$$x(\delta^+(j)) = 1 \qquad \forall\, j \in V^{fe} \tag{3.21}$$

$$x(\delta^+(j)) \geq 1 \qquad \forall\, j \in V^{a} \tag{3.22}$$

$$x(\delta^+(j)) \geq y_j \qquad \forall\, j \in V^{pa} \tag{3.23}$$

$$x(\delta^+(j)) = y_j \qquad \forall\, j \in V^{e} \tag{3.24}$$

$$x(\delta^+(j)) - x(\delta^-(j)) = 0 \qquad \forall\, j \in V'' \tag{3.25}$$

$$y_{i_k} \geq y_{i_{k+1}} \qquad \forall\, i \in I, k \in \{1, \ldots, \bar{\beta}_i\} \tag{3.26}$$

$$\sum_{k \in V_i} g_k = d_i \qquad \forall\, i \in I \tag{3.27}$$

$$x(\bar{S} : S) \geq max\left(\frac{-\sum_{i \in S} g_i}{Q}, 1\right) \qquad \forall\, S \subseteq V'' \setminus \{0\}, \bar{S} = V'' \setminus S \setminus \{0\} \tag{3.28}$$

$$y_i \leq g_i \leq d_{I_i} y_i \qquad \forall\, i \in V'' \setminus \{0\} : d_{I_i} \geq 0 \tag{3.29}$$

$$d_{I_i} y_i \leq g_i \leq -y_i \qquad \forall\, i \in V'' \setminus \{0\} : d_{I_i} < 0 \tag{3.30}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i,j \in V'' : i \in V^e \cup V^{fe} \text{ or } j \in V^e \cup V^{fe} \tag{3.31}$$

$$x_{ij} \in \{0,1,..,u_{I_i I_j}\} \qquad \forall\, i,j \in V^a \cup V^{pa} \tag{3.32}$$

$$y_i \in \{0,1\} \qquad \forall\, i \in V'' \setminus \{0\} \tag{3.33}$$

$$g_i \lesseqgtr 0, integer \qquad \forall\, i \in V'' \setminus \{0\} \tag{3.34}$$

The objective function (3.19) minimizes the total distance travelled. Constraints (3.20)-(3.24) define the degree of each vertex. Constraints (3.25) specify flow conservation for the vehicle and (3.26) impose the order for the usage of the duplicates. The total demand performed at each station must be equal to the required, amount as stated by (3.27), and the vehicle capacity must not be exceeded at any point, as stated by (3.28). Notice that constraints (3.28) also serve the purpose of preventing subtours. Constraints (3.29) and (3.30) specify lower and upper bounds for the g variables. Finally constraints (3.31)-(3.34) define the formulation variables.

This formulation has some interesting properties:

PROPERTY 13 *Formulation GF is equivalent to the relaxation AF when each station is represented by a single vertex.*

Proof In case $V'' = V^a \cup \{0\}$ and $V^{pa} = V^{fe} = V^e = \emptyset$, each station is represented by a single aggregated vertex, which corresponds to the scenario in formulation AF. Then all constraints associated to vertices other than those in V^a are removed. In addition, notice that in this case $g_j = d_{I_j}$ for each $j \in V''$. Consequently, constraints (3.28) are equivalent to (3.6) and constraints (3.29) and (3.30) might be removed as they become redundant. □

PROPERTY 14 *In case each station $i \in I$ is represented by β_i duplicates in G'', solving formulation GF always results in the optimal solution for the SBRPW.*

Proof Recall that the shortcomings that might render an aggregated solution infeasible may only exist when there is at least one vertex being visited multiple times. By representing each station $i \in I$ with β_i duplicates we obtain the extended network, where each vertex is visited at most once. Therefore, solving formulation GF under this network always produces a feasible solution. □

Because of the exponentially-many capacity constraints (3.28) we solve formulation GF using a B&C algorithm. These constraints are separated using the same procedure described in Chemla et al. (2013), which is based on the evaluation of a maximum flow on a support graph. Although in classical B&C algorithms from the literature, the separation procedure is invoked at each node of the enumeration tree, we found it convenient to separate cuts only at integer nodes (using the so-called lazy callback). This decision is based on the significant developments in commercial MILP solvers in recent years, which have evolved up to a point where it might be more efficient to use this approach instead of the classical one.

Now, let us propose an exact algorithm called the *minimal extended network* (MEN) based on the following two observations. Firstly, most often than not, in optimal solutions for the SBRPW stations are visited no more than 2 or 3 times. This behavior was already hinted by Salazar-González and Santos-Hernández (2015) while solving the SD1PDTSP, which is a closely related problem. Secondly, by performing extensive computational experiments with formulation AF, we noticed that often enough the solution found corresponds to the optimal solution for the SBRPW.

Basically, the algorithm starts by solving formulation GF over the aggregated scenario, where each station is represented by a single vertex. A feasibility check is then performed with the solution found and in case the solution is indeed feasible, MEN terminates with an optimal SBRPW solution. Otherwise, we inspect the solution in order to determine which vertices are being visited more than once. Recall that $\bar{\beta}_i$ specify the number of visits to vertex i in a solution. Each vertex $i \in V''$, such that $\bar{\beta}_i > 1$, is split into $\bar{\beta}_i$ duplicates and then, the next iteration solves formulation GF under the modified graph. According to the observation presented in Section 3.3, this split operation guarantees that the existent infeasibilities are removed. The procedure iterates until the optimal solution found is feasible for the SBRPW.

Furthermore, at each iteration of MEN a list of the best 10 incumbent solutions found is kept. In case the solution found is infeasible, this list is checked as an attempt of finding a feasible upper bound. The incumbent upper bound is then used as a warm start for the next iterations.

During computational experiments we notice that, for several instances, the first iteration of MEN produces an infeasible aggregated solution $\bar{x}$, but the reverse solution $\tilde{x}$ such that $\tilde{x}_{ij} = \bar{x}_{ji}$, is actually feasible. To take advantage of this observation an additional feasibility test is performed in the reverse solution of the first iteration if needed.

PROPERTY 15 *Algorithm MEN converges to the optimal solution of the SBRPW.*

Proof In the worst case scenario, MEN iterates for $\sum_{i \in I} \beta_i$ times, performing a single split at each iteration and culminating in the extended network in the last iteration. Therefore, because of Property 14, it is guaranteed that MEN produces the optimal solution for the SBRPW. □

According to Property 15, algorithm MEN might be as time consuming (or even more) as directly solving the extended network. However, computational experiments indicate this is not the case and, in practice, most often than not only a few iterations are necessary to converge to an optimal SBRPW solution.

Furthermore, notice that at the end of each iteration a feasibility test is necessary to determine whether MEN should stop or continue to the next iteration. In the following, we propose a procedure, called *split-and-inspect*, which is used by MEN to verify if a given aggregated solution is feasible for the SBRPW.

3.5.1 Split-and-inspect

Let $(\bar{z}, \bar{x}, \bar{g})$ be a solution of formulation GF having value $\bar{z}$, consider $\bar{V}^{+1} = \{i : i \in V'', \bar{\beta}_i > 1\}$ and $\bar{V}^1 = V'' \setminus \bar{V}^{+1}$. Then, define $\tilde{G} = (\tilde{V}, \tilde{A})$ as a support graph obtained by splitting each vertex $i \in \bar{V}^{+1}$ into $\bar{\beta}_i$ duplicates, and including all vertices in $\bar{V}^1$ directly into $\tilde{V}$. Let $\sigma(i)$ specify the vertex in V'' associated with $i \in \tilde{V}$.

We now propose the following formulation to verify whether or not $(\bar{z}, \bar{x}, \bar{g})$ is induced by a disaggregated feasible solution for the SBRPW. This formulation uses the same variables x and g as in GF, but defined over $\tilde{G}$.

$$\text{(SIF)} \quad \min\ 0 \tag{3.35}$$

subject to

$$(3.25), (3.27) - (3.30)$$

$$\sum_{i \in \tilde{V}} \sum_{j \in \tilde{V}} c_{ij} x_{ij} = \bar{z} \tag{3.36}$$

$$\sum_{i \in \tilde{V}} x_{ij} = \sum_{i \in \tilde{V}} \bar{x}_{ij} \qquad \forall\ j \in \bar{V}^1 \tag{3.37}$$

$$\sum_{i \in \tilde{V}} x_{ij} = 1 \qquad \forall\ j \in V'' \setminus \bar{V}^1 \tag{3.38}$$

$$x_{ij} = \bar{x}_{ij} \qquad \forall\ i, j \in \bar{V}^1 \tag{3.39}$$

$$x_{ij} = 0 \qquad \forall\ i, j \in \tilde{V} : \{\sigma(i), \sigma(j)\} \cap \bar{V}^{+1} \neq \emptyset, \bar{x}_{\sigma(i)\sigma(j)} = 0 \tag{3.40}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\ i, j \in \tilde{V} \tag{3.41}$$

$$g_i \leqslant 0, integer \qquad \forall\ i \in \tilde{V} \tag{3.42}$$

Constraints (3.36) make sure that the search is performed for a disaggregated solution of the same value $\bar{z}$ of the original aggregated one, whereas (3.37)-(3.38) specify that stations must be visited the same number of times as in the original solution. Arcs between unmodified vertices keep the same values, as stated by (3.39), and based on the values $\bar{x}$, constraints (3.40) remove several arcs that should not be used.

In case formulation SIF returns a feasible solution, then the corresponding aggregated solution is also feasible and the associated route may be retrieved by simply inspecting the solution of SIF.

3.6 Computational experiments

Our algorithms were implemented in C++ and computational experiments were run on a PC with an Intel Core i7-3770 3.40 GHz with 8 Gb of RAM. We used Cplex 12.6.2 to solve the formulations and to implement the B&C algorithms. The default options of Cplex were

selected, but the use of a single thread was imposed. The algorithms were given a time limit of 1 hour for each run.

In order to test our algorithms, we modified the first set of benchmark instances proposed by Chemla et al. (2013) for the SBRP, and later used by Erdoğan et al. (2015). During the creation of theses instances, distance values are rounded down to the nearest integer and, thus, triangle inequalities are not satisfied. Then, by running Algorithm 2 on each instance of this set we derived a new set of benchmark instances, where the triangle inequalities hold. Notice that, in Algorithm 2 *distance* refers to a symmetric array of distance values and *size* is the number of vertices, i.e., stations and the depot. The new instances are available at `http://www.or.unimore.it/resources/`.

Algorithm 2 Algorithm to impose the triangle inequalities

```
1: function imposeTriangleInequalities(distances, size)
2:     done ← false
3:     i; j; k
4:     while not done do
5:         done ← true
6:         for i ← 0 to size − 1 do
7:             for j ← 0 to size − 1 do
8:                 for k ← 0 to size − 1 such that k ≠ i and k ≠ j do
9:                     if distances[i][k] > distances[i][j] + distances[j][k] then
10:                        distances[i][k] ← distances[i][j] + distances[j][k]
11:                        distances[k][i] ← distances[k][j] + distances[j][i]
12:                        done ← false
13:                    end if
14:    end while
15: end function
```

In general, the proposed *split-and-inspect* procedure to verify the feasibility of aggregated solutions is quite fast, however when there are several stations being visited many times the algorithm may becomes inefficient, as it tends to degrade to solving the extended network.

As mentioned in Section 3.5, in optimal solutions, most often than not only a few stations are visited multiple times, and the number of visits is usually not greater than three. This confirms the results in Salazar-González and Santos-Hernández (2015). Indeed, because in our algorithm MEN the feasibility test is only run at the end of iterations and the solutions checked are usually of good quality, during our experiments no significant efficiency issues were encounter with the *split-and-inspect* procedure. However, this was not the case for the BinArc formulation due to the fact that, at the beginning of the branch-and-bound sometimes the first incumbent solutions generated are of bad quality and may contain several stations being visited multiple times, which requires a great effort from our feasibility check. This

may be easily fixed by providing the formulation with a valid upper bound, used as warm start. To this end we adapted the *iterated local search* (ILS) algorithm proposed by Cruz et al. (2016) for the SBRP to provide valid upper bounds not only for the BinArc formulation but also for MEN algorithm.

In Table 3.1 we present aggregated results from extensive computational experiments performed with our benchmark instances. We tested formulation BinArc giving an initial upper bound and the MEN algorithm with and without an initial upper bound. Results are aggregated per instance size. The table provides the following information: number of proved optimal solutions (*opt*), average percentage gaps between lower and upper bounds (*avg gap*), and average number of seconds required by the respective algorithm (*avg sec*). Formulation BinArc is clearly outperformed by algorithm MEN, finding only 278 proved optimal solutions whereas MEN, with and without an initial upper bound, is able to find 397 and 402 optimal solutions, respectively. In addition, notice that formulation BinArc requires in average about 3 times more time than MEN. A curious and surprising behavior of MEN is the fact that giving an initial upper bound did not improve the average efficiency of the algorithm. In fact, for several instances the opposite behavior was observed. This might be explained by the observation that formulation *GF* usually finds feasible upper bounds quite fast but takes longer to improve its lower bound value and by giving a warm start the branching decision taken by Cplex are not the same. Indeed, Fischetti and Monaci (2014), and Lodi and Tramontani (2013) show that changes in the initial conditions in mixed integer programming softwares, may lead to significant changes in the behavior of the solver and its efficiency.

Table 3.1: Aggregated results for the benchmark set.

		BinArc + initial UB			MEN + initial UB			MEN		
size	#	opt	avg gap	avg sec	opt	avg gap	avg sec	opt	avg gap	avg sec
n=20	90	86	0.17	166.89	90	0.00	2.78	90	0.00	4.21
n=30	90	83	0.12	460.30	88	0.09	82.02	88	0.05	81.38
n=40	90	48	0.86	1912.01	83	0.39	406.57	84	0.21	308.66
n=50	90	36	2.97	2330.94	71	1.42	851.30	71	1.23	806.09
n=60	90	25	4.33	2729.89	65	2.29	1075.94	69	1.06	963.54
sum/avg	450	278	1.69	1520.01	397	0.84	483.72	402	0.51	432.78

In Tables 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10 and 3.11 we present the detailed results of the experiments reported in Table 3.1. The value of the upper bound solution given to formulation BinArc as a warm start is reported in column *init*. Notice that this upper bound is the same given to *MEN + initial UB*. Columns named *ub* specify the best feasible upper bound found by the corresponding algorithm, and *gap* specify the difference between the value of the best upper and lower bounds found. Finally, column *sec* reports the required computational time in seconds. For simplicity, when the time limit is reached the value *t.lim.*

is used instead.

As expected, instances in which the vehicle capacity is tighter are more difficult to solve, specially when the number of stations is either 50 or 60 and $Q = 10$, where all three approaches struggle to converge to a proved optimal SBRPW solution. In addition, notice that increasing the vehicle capacity no longer has any effects on the solution cost after a certain threshold, which tends to range around $Q = 40$ for several instances of our benchmark set. Furthermore, we report that the average computational time required for each call to the *split-and-inspect* procedure was usually not greater than a second, which seems to attest its efficiency in this scenario.

3.7 Conclusions

In this chapter we introduced the SBRPW, presented a few interesting theoretical results and proposed two exact algorithms to solve the problem. The first approach, called BinArc formulation, is based on an interesting strategy that allows us to have binary variables on arcs instead of integer ones, enabling the use of infeasible path constraints to remove solutions with temporary operations. The second approach, namely MEN, solves a relaxed aggregated formulation at each iteration and aims at solving the SBRPW by duplicating as few vertices as possible.

We derived a new set of 450 benchmark instances and performed extensive computational experiments with our approaches. Although formulation BinArc is outmatched by MEN, it was able to find 278 proved optimal solutions, while with algorithm MEN we were able to prove 402 optimal solutions in the best configuration. Furthermore, the results also show that the proposed feasibility test, based on a ILP formulation, is quite efficient in this scenario.

Table 3.2: Results for instances with 20 stations and $q = \{10, 15, 20, 25, 30\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n20A	10	4702	4702	—	0.02	4702	—	0.62	4702	—	0.20
n20B	10	4769	4769	—	0.01	4769	—	0.01	4769	—	0.04
n20C	10	6025	6025	—	0.01	6025	—	0.29	6025	—	182.09
n20D	10	6043	6043	—	0.01	6043	—	0.01	6043	—	0.15
n20E	10	6245	6245	—	0.02	6245	—	0.02	6245	—	0.30
n20F	10	4799	4799	—	0.01	4799	—	0.14	4799	—	0.73
n20G	10	5070	5070	—	0.01	5070	—	0.01	5070	—	0.06
n20H	10	5542	5542	—	0.02	5542	—	0.02	5542	—	0.14
n20I	10	4590	4590	—	0.02	4590	—	0.02	4590	—	0.08
n20J	10	4193	4193	—	0.02	4193	—	0.02	4193	—	0.42
n20A	15	3948	3948	—	0.01	3948	—	0.01	3948	—	0.06
n20B	15	4174	4174	—	0.01	4174	—	0.01	4174	—	0.02
n20C	15	5137	5137	—	0.02	5137	—	0.02	5137	—	0.42
n20D	15	5488	5488	—	0.01	5488	—	0.01	5488	—	0.19
n20E	15	5560	5560	—	0.02	5560	—	0.02	5560	—	0.29
n20F	15	4492	4492	—	0.01	4492	—	0.01	4492	—	0.11
n20G	15	4567	4567	—	0.01	4567	—	0.01	4567	—	0.09
n20H	15	4745	4745	—	0.01	4745	—	0.01	4745	—	0.06
n20I	15	4205	4183	—	0.01	4183	—	0.11	4183	—	0.10
n20J	15	3947	3947	—	0.02	3947	—	0.01	3947	—	0.04
n20A	20	3585	3585	—	0.01	3585	—	0.01	3585	—	0.01
n20B	20	4073	4073	—	0.01	4073	—	0.01	4073	—	0.02
n20C	20	4486	4486	—	0.01	4486	—	0.01	4486	—	0.04
n20D	20	4703	4703	—	0.01	4703	—	0.02	4703	—	0.47
n20E	20	4599	4599	—	0.01	4599	—	0.01	4599	—	0.12
n20F	20	4252	4252	—	0.01	4252	—	0.13	4252	—	0.35
n20G	20	4354	4354	—	0.01	4354	—	0.01	4354	—	0.06
n20H	20	4364	4364	—	0.01	4364	—	0.01	4364	—	0.10
n20I	20	4071	4071	—	0.01	4071	—	0.01	4071	—	0.08
n20J	20	3787	3787	—	0.01	3787	—	0.01	3787	—	0.07
n20A	25	3583	3583	—	0.01	3583	—	0.01	3583	—	0.01
n20B	25	3792	3792	—	0.01	3792	—	0.01	3792	—	0.01
n20C	25	4189	4189	—	0.01	4189	—	0.01	4189	—	0.01
n20D	25	4327	4327	—	0.01	4327	—	0.12	4327	—	0.39
n20E	25	4556	4556	—	0.01	4556	—	0.01	4556	—	0.04
n20F	25	4252	4252	—	0.01	4252	—	0.13	4252	—	0.37
n20G	25	4216	4216	—	0.01	4216	—	0.01	4216	—	0.03
n20H	25	4041	4041	—	0.02	4041	—	0.01	4041	—	0.02
n20I	25	4053	4053	—	0.01	4053	—	0.01	4053	—	0.07
n20J	25	3763	3763	—	0.01	3763	—	0.01	3763	—	0.02
n20A	30	3583	3583	—	0.01	3583	—	0.01	3583	—	0.01
n20B	30	3792	3792	—	0.01	3792	—	0.01	3792	—	0.02
n20C	30	4184	4184	—	0.01	4184	—	0.01	4184	—	0.05
n20D	30	4140	4140	—	0.01	4140	—	0.01	4140	—	0.05
n20E	30	4388	4388	—	0.01	4388	—	0.01	4388	—	0.28
n20F	30	4252	4252	—	0.02	4252	—	0.14	4252	—	0.37
n20G	30	4216	4216	—	0.01	4216	—	0.01	4216	—	0.03
n20H	30	4041	4041	—	0.01	4041	—	0.01	4041	—	0.03
n20I	30	3996	3996	—	0.01	3996	—	0.01	3996	—	0.04
n20J	30	3763	3763	—	0.01	3763	—	0.01	3763	—	0.02

Table 3.3: Results for instances with 20 stations and $q = \{35, 40, 45, 1000\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n20A	35	3583	3583	—	0.01	3583	—	0.01	3583	—	0.01
n20B	35	3792	3792	—	0.01	3792	—	0.01	3792	—	0.01
n20C	35	4045	4045	—	0.01	4045	—	0.01	4045	—	0.04
n20D	35	4122	4122	3.84	0.02	4122	—	1.90	4122	—	49.89
n20E	35	4277	4277	—	0.01	4277	—	0.01	4277	—	0.03
n20F	35	4252	4252	—	0.01	4252	—	0.12	4252	—	0.37
n20G	35	4216	4216	—	0.01	4216	—	0.01	4216	—	0.03
n20H	35	4041	4041	—	0.02	4041	—	0.01	4041	—	0.02
n20I	35	3996	3996	—	0.01	3996	—	0.01	3996	—	0.04
n20J	35	3763	3763	—	0.02	3763	—	0.01	3763	—	0.02
n20A	40	3583	3583	—	0.01	3583	—	0.01	3583	—	0.01
n20B	40	3792	3792	—	0.01	3792	—	0.01	3792	—	0.01
n20C	40	4042	4042	—	0.01	4042	—	0.01	4042	—	0.04
n20D	40	4121	4121	3.76	0.01	4121	—	1.58	4121	—	36.03
n20E	40	4277	4277	—	0.01	4277	—	0.01	4277	—	0.03
n20F	40	4252	4252	—	0.01	4252	—	0.13	4252	—	0.37
n20G	40	4216	4216	—	0.01	4216	—	0.01	4216	—	0.03
n20H	40	4041	4041	—	0.02	4041	—	0.02	4041	—	0.02
n20I	40	3996	3996	—	0.01	3996	—	0.01	3996	—	0.04
n20J	40	3763	3763	—	0.02	3763	—	0.02	3763	—	0.02
n20A	45	3583	3583	—	0.01	3583	—	0.01	3583	—	0.01
n20B	45	3792	3792	—	0.01	3792	—	0.01	3792	—	0.01
n20C	45	4042	4042	—	0.01	4042	—	0.01	4042	—	0.05
n20D	45	4121	4121	3.83	0.01	4121	—	1.51	4121	—	40.69
n20E	45	4277	4277	—	0.01	4277	—	0.02	4277	—	0.03
n20F	45	4252	4252	—	0.01	4252	—	0.13	4252	—	0.38
n20G	45	4216	4216	—	0.01	4216	—	0.01	4216	—	0.03
n20H	45	4041	4041	—	0.02	4041	—	0.01	4041	—	0.02
n20I	45	3996	3996	—	0.01	3996	—	0.01	3996	—	0.04
n20J	45	3763	3763	—	0.01	3763	—	0.01	3763	—	0.02
n20A	1000	3583	3583	—	0.01	3583	—	0.01	3583	—	0.01
n20B	1000	3792	3792	—	0.01	3792	—	0.01	3792	—	0.02
n20C	1000	4042	4042	—	0.01	4042	—	0.02	4042	—	0.05
n20D	1000	4121	4121	3.85	0.01	4121	—	1.91	4121	—	60.96
n20E	1000	4277	4277	—	0.02	4277	—	0.02	4277	—	0.03
n20F	1000	4252	4252	—	0.01	4252	—	0.13	4252	—	0.39
n20G	1000	4216	4216	—	0.01	4216	—	0.01	4216	—	0.03
n20H	1000	4041	4041	—	0.01	4041	—	0.01	4041	—	0.02
n20I	1000	3996	3996	—	0.01	3996	—	0.01	3996	—	0.04
n20J	1000	3763	3763	—	0.01	3763	—	0.01	3763	—	0.02

Table 3.4: Results for instances with 30 stations and $q = \{10, 15, 20, 25, 30\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n30A	10	6236	6236	—	151.82	6236	—	65.10	6236	—	11.84
n30B	10	6315	6315	1.46	t.lim.	6315	—	1.76	6315	—	1.12
n30C	10	6335	6335	—	26.20	6335	—	25.46	6335	—	10.82
n30D	10	6138	6076	—	38.67	6076	—	17.73	6076	—	6.69
n30E	10	5883	5883	—	274.50	5883	—	1.51	5883	—	0.90
n30F	10	5695	5695	—	1.97	5695	—	0.71	5695	—	0.92
n30G	10	8911	8891	—	594.35	8911	4.94	t.lim.	8926	1.57	t.lim.
n30H	10	5951	5951	—	1043.15	5951	—	24.91	5951	—	2.58
n30I	10	5564	5430	—	22.26	5430	—	1.96	5430	—	1.48
n30J	10	5764	5764	—	6.39	5764	—	1.57	5764	—	17.22
n30A	15	5442	5442	—	4.36	5442	—	2.93	5442	—	0.87
n30B	15	5376	5376	—	23.85	5376	—	0.94	5376	—	0.68
n30C	15	5170	5170	—	0.67	5170	—	0.15	5170	—	0.36
n30D	15	5249	5142	—	3.23	5142	—	0.47	5142	—	0.26
n30E	15	5121	5121	—	69.49	5121	—	0.16	5121	—	0.17
n30F	15	4897	4897	—	0.73	4897	—	0.14	4897	—	0.11
n30G	15	7229	7221	4.66	t.lim.	7148	3.04	t.lim.	7148	2.92	t.lim.
n30H	15	5123	5123	—	7.61	5123	—	1.46	5123	—	15.71
n30I	15	4622	4622	—	0.71	4622	—	0.36	4622	—	0.23
n30J	15	5249	5249	—	3.96	5249	—	0.48	5249	—	5.90
n30A	20	4940	4911	—	1.43	4911	—	0.37	4911	—	0.98
n30B	20	4831	4831	—	1.57	4831	—	0.26	4831	—	0.34
n30C	20	4831	4831	—	0.51	4831	—	0.16	4831	—	0.18
n30D	20	4926	4871	—	0.60	4871	—	0.35	4871	—	0.36
n30E	20	4675	4675	—	4.44	4675	—	0.18	4675	—	0.10
n30F	20	4439	4439	—	0.60	4439	—	0.10	4439	—	0.05
n30G	20	6399	6399	0.22	t.lim.	6399	—	7.72	6399	—	25.91
n30H	20	4275	4275	—	0.39	4275	—	0.12	4275	—	0.21
n30I	20	4325	4325	—	0.17	4325	—	0.04	4325	—	0.03
n30J	20	4620	4620	—	196.61	4620	—	0.18	4620	—	0.09
n30A	25	4825	4825	—	4.29	4825	—	0.21	4825	—	0.11
n30B	25	4583	4583	—	0.61	4583	—	0.16	4583	—	0.13
n30C	25	4586	4586	—	5.46	4586	—	0.19	4586	—	0.14
n30D	25	4723	4723	—	0.25	4723	—	0.12	4723	—	0.07
n30E	25	4598	4598	—	6.27	4598	—	0.11	4598	—	0.10
n30F	25	4439	4439	—	0.61	4439	—	0.07	4439	—	0.04
n30G	25	5731	5731	—	7.98	5731	—	2.13	5731	—	0.75
n30H	25	4163	4163	—	0.60	4163	—	0.12	4163	—	0.08
n30I	25	4272	4272	—	0.09	4272	—	0.06	4272	—	0.06
n30J	25	4549	4549	—	28.12	4549	—	0.16	4549	—	0.07
n30A	30	4731	4731	—	272.70	4731	—	0.86	4731	—	0.65
n30B	30	4583	4583	—	15.07	4583	—	0.19	4583	—	0.13
n30C	30	4565	4565	—	24.26	4565	—	0.17	4565	—	0.09
n30D	30	4701	4701	—	0.71	4701	—	1.07	4701	—	0.56
n30E	30	4588	4588	—	3.38	4588	—	0.10	4588	—	0.08
n30F	30	4417	4417	—	1.13	4417	—	0.08	4417	—	0.46
n30G	30	5439	5439	—	8.07	5439	—	1.39	5439	—	0.45
n30H	30	4163	4163	—	171.00	4163	—	0.61	4163	—	0.77
n30I	30	4272	4272	—	0.28	4272	—	0.06	4272	—	0.06
n30J	30	4384	4384	—	10.08	4384	—	0.14	4384	—	0.43

Table 3.5: Results for instances with 30 stations and $q = \{35, 40, 45, 1000\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n30A	35	4660	4660	—	4.71	4660	—	0.11	4660	—	0.09
n30B	35	4583	4583	1.36	t.lim.	4583	—	0.13	4583	—	0.13
n30C	35	4525	4525	—	70.64	4525	—	0.14	4525	—	0.08
n30D	35	4696	4696	—	7.79	4696	—	0.13	4696	—	0.10
n30E	35	4588	4588	—	3.18	4588	—	0.12	4588	—	0.08
n30F	35	4411	4411	—	1.36	4411	—	0.13	4411	—	0.05
n30G	35	5211	5211	—	1.38	5211	—	0.41	5211	—	0.26
n30H	35	4163	4163	—	412.45	4163	—	0.64	4163	—	0.75
n30I	35	4272	4272	—	0.27	4272	—	0.08	4272	—	0.06
n30J	35	4372	4372	—	4.42	4372	—	0.13	4372	—	0.07
n30A	40	4660	4660	—	4.65	4660	—	0.10	4660	—	0.09
n30B	40	4531	4531	—	3012.23	4531	—	0.12	4531	—	0.11
n30C	40	4475	4475	—	463.36	4475	—	0.12	4475	—	0.63
n30D	40	4696	4696	—	1625.74	4696	—	0.72	4696	—	0.63
n30E	40	4588	4588	—	2.49	4588	—	0.09	4588	—	0.08
n30F	40	4411	4411	—	1.38	4411	—	0.13	4411	—	0.05
n30G	40	5031	5031	—	3091.69	5031	—	0.36	5031	—	0.16
n30H	40	4163	4163	—	411.75	4163	—	0.75	4163	—	0.75
n30I	40	4272	4272	—	0.28	4272	—	0.05	4272	—	0.06
n30J	40	4372	4372	—	19.39	4372	—	0.87	4372	—	0.99
n30A	45	4660	4660	—	4.75	4660	—	0.09	4660	—	0.09
n30B	45	4531	4531	1.22	t.lim.	4531	—	0.10	4531	—	0.11
n30C	45	4475	4475	—	465.91	4475	—	0.09	4475	—	0.64
n30D	45	4696	4696	—	2193.45	4696	—	0.55	4696	—	0.54
n30E	45	4588	4588	—	3.30	4588	—	0.10	4588	—	0.08
n30F	45	4411	4411	—	1.32	4411	—	0.10	4411	—	0.05
n30G	45	4827	4827	—	0.44	4827	—	0.16	4827	—	0.14
n30H	45	4163	4163	—	408.35	4163	—	0.64	4163	—	0.74
n30I	45	4272	4272	—	0.29	4272	—	0.08	4272	—	0.06
n30J	45	4372	4372	—	19.44	4372	—	0.55	4372	—	1.02
n30A	1000	4660	4660	—	4.71	4660	—	0.13	4660	—	0.09
n30B	1000	4531	4531	1.46	t.lim.	4531	—	0.14	4531	—	0.11
n30C	1000	4475	4475	—	463.69	4475	—	0.13	4475	—	0.66
n30D	1000	4696	4696	0.45	t.lim.	4696	—	1.04	4696	—	0.69
n30E	1000	4588	4588	—	3.41	4588	—	0.12	4588	—	0.08
n30F	1000	4411	4411	—	1.34	4411	—	0.10	4411	—	0.05
n30G	1000	4781	4781	—	1.52	4781	—	0.08	4781	—	0.07
n30H	1000	4163	4163	—	412.41	4163	—	0.84	4163	—	0.84
n30I	1000	4272	4272	—	0.28	4272	—	0.06	4272	—	0.07
n30J	1000	4372	4372	—	19.18	4372	—	0.55	4372	—	1.08

Table 3.6: Results for instances with 40 stations and $q = \{10, 15, 20, 25, 30\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n40A	10	7132	7132	3.88	t.lim.	7132	2.86	t.lim.	—	—	t.lim.
n40B	10	5978	5978	—	94.24	5978	—	2415.63	5978	—	97.09
n40C	10	7237	7237	—	732.53	7237	2.04	t.lim.	7237	—	12.56
n40D	10	7967	7820	3.26	t.lim.	7967	12.99	t.lim.	7794	—	15.50
n40E	10	6660	6647	6.32	t.lim.	6636	2.90	t.lim.	6610	2.83	t.lim.
n40F	10	7170	7108	1.84	t.lim.	7170	4.02	t.lim.	7084	—	1320.98
n40G	10	7391	7391	4.37	1636.34	7375	1.22	t.lim.	—	—	t.lim.
n40H	10	6629	6584	—	2780.40	6629	9.16	t.lim.	6584	9.08	t.lim.
n40I	10	6960	6960	2.11	t.lim.	6950	—	1957.12	7000	4.29	2475.48
n40J	10	6354	6354	1.14	t.lim.	6354	—	1737.17	6355	2.16	t.lim.
n40A	15	6107	6107	—	854.54	6107	—	153.43	6107	—	13.81
n40B	15	5350	5350	—	43.42	5350	—	3.55	5350	—	0.58
n40C	15	5916	5916	1.71	t.lim.	5916	—	8.33	5916	—	6.62
n40D	15	6466	6465	2.01	t.lim.	6465	—	4.12	6465	—	798.23
n40E	15	5776	5776	0.76	t.lim.	5776	—	31.23	5776	—	23.88
n40F	15	5916	5916	0.43	t.lim.	5916	—	132.08	5916	—	46.60
n40G	15	6426	6426	—	149.38	6426	—	663.28	6426	—	1536.10
n40H	15	5578	5578	—	7.89	5578	—	1.67	5578	—	13.21
n40I	15	5781	5781	—	43.34	5781	—	1.48	5781	—	4.28
n40J	15	5655	5655	—	41.19	5655	—	8.01	5655	—	6.90
n40A	20	5449	5449	—	9.10	5449	—	1.78	5449	—	2.22
n40B	20	5310	5310	1.93	t.lim.	5310	—	2.64	5310	—	2.42
n40C	20	5349	5347	—	35.71	5347	—	3138.41	5347	—	2195.59
n40D	20	5952	5952	1.52	t.lim.	5952	—	240.72	5952	—	330.25
n40E	20	5505	5505	3.29	t.lim.	5505	—	673.05	5505	—	659.54
n40F	20	5289	5289	—	5.62	5289	—	1.84	5289	—	0.99
n40G	20	5476	5476	—	1.27	5476	—	0.35	5476	—	0.22
n40H	20	5253	5253	1.61	t.lim.	5253	—	7.13	5253	—	8.90
n40I	20	5109	5109	—	10.82	5109	—	0.30	5109	—	0.48
n40J	20	5121	5121	0.97	t.lim.	5121	—	1.76	5121	—	1.02
n40A	25	5097	5052	—	403.33	5052	—	1.08	5052	—	2.29
n40B	25	5253	5253	2.08	t.lim.	5253	—	0.38	5253	—	0.42
n40C	25	5131	5131	1.33	t.lim.	5131	—	3.88	5131	—	3.13
n40D	25	5853	5853	1.37	t.lim.	5853	—	37.36	5853	—	36.27
n40E	25	5392	5392	1.70	t.lim.	5392	—	22.86	5392	—	26.11
n40F	25	5176	5176	0.29	t.lim.	5176	—	11.81	5176	—	11.70
n40G	25	5361	5361	—	0.92	5361	—	0.16	5361	—	0.13
n40H	25	4981	4981	—	466.11	4981	—	0.22	4981	—	0.20
n40I	25	5018	5018	1.13	t.lim.	5018	—	4.06	5018	—	4.44
n40J	25	4869	4869	—	10.36	4869	—	0.18	4869	—	0.44
n40A	30	5042	5042	—	425.74	5042	—	0.16	5042	—	0.20
n40B	30	5253	5253	1.90	t.lim.	5253	—	0.31	5253	—	0.26
n40C	30	4780	4780	1.63	t.lim.	4780	—	0.26	4780	—	0.28
n40D	30	5480	5480	0.83	t.lim.	5480	—	0.38	5480	—	0.36
n40E	30	5279	5279	—	301.33	5279	—	3.50	5279	—	3.27
n40F	30	5049	5049	—	324.44	5049	—	5.15	5049	—	2.25
n40G	30	5312	5312	—	0.65	5312	—	0.16	5312	—	0.21
n40H	30	4939	4939	—	1798.44	4939	—	9.17	4939	—	8.64
n40I	30	4912	4912	0.81	t.lim.	4912	—	3.69	4912	—	2.82
n40J	30	4811	4811	—	3.28	4811	—	0.15	4811	—	0.20

Table 3.7: Results for instances with 40 stations and $q = \{35, 40, 45, 1000\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n40A	35	5042	5042	—	282.07	5042	—	0.23	5042	—	0.18
n40B	35	5253	5253	2.07	t.lim.	5253	—	0.52	5253	—	0.33
n40C	35	4684	4684	2.26	t.lim.	4684	—	2.28	4684	—	1.70
n40D	35	5340	5340	—	2138.01	5340	—	1.18	5340	—	1.08
n40E	35	5210	5210	—	104.03	5210	—	1.44	5210	—	2.16
n40F	35	5007	5007	—	56.21	5007	—	3.68	5007	—	2.90
n40G	35	5312	5312	—	5.79	5312	—	0.18	5312	—	0.17
n40H	35	4939	4939	1.10	t.lim.	4939	—	10.46	4939	—	10.55
n40I	35	4912	4912	0.67	t.lim.	4912	—	3.31	4912	—	4.02
n40J	35	4811	4811	—	3.42	4811	—	0.11	4811	—	0.20
n40A	40	5042	5042	—	283.74	5042	—	0.16	5042	—	0.17
n40B	40	5253	5253	1.91	t.lim.	5253	—	0.32	5253	—	0.34
n40C	40	4684	4684	2.58	t.lim.	4684	—	1.32	4684	—	1.69
n40D	40	5304	5304	—	3096.48	5304	—	1.08	5304	—	2.01
n40E	40	5209	5209	—	100.08	5209	—	1.25	5209	—	1.33
n40F	40	4944	4944	—	468.65	4944	—	4.23	4944	—	3.77
n40G	40	5312	5312	—	6.13	5312	—	0.16	5312	—	0.18
n40H	40	4939	4939	1.67	t.lim.	4939	—	12.44	4939	—	11.81
n40I	40	4912	4912	0.69	t.lim.	4912	—	2.72	4912	—	3.47
n40J	40	4811	4811	—	3.42	4811	—	0.11	4811	—	0.20
n40A	45	5042	5042	—	284.19	5042	—	0.17	5042	—	0.17
n40B	45	5253	5253	1.96	t.lim.	5253	—	0.31	5253	—	0.36
n40C	45	4684	4684	2.58	t.lim.	4684	—	1.94	4684	—	1.73
n40D	45	5295	5295	—	1934.82	5295	—	0.23	5295	—	0.13
n40E	45	5209	5209	—	125.68	5209	—	1.84	5209	—	1.34
n40F	45	4944	4944	—	609.20	4944	—	6.17	4944	—	4.21
n40G	45	5312	5312	—	7.95	5312	—	0.16	5312	—	0.17
n40H	45	4939	4939	1.94	t.lim.	4939	—	11.85	4939	—	11.28
n40I	45	4912	4912	0.73	t.lim.	4912	—	3.29	4912	—	4.02
n40J	45	4811	4811	—	7.18	4811	—	0.10	4811	—	0.20
n40A	1000	5042	5042	—	431.59	5042	—	0.16	5042	—	0.16
n40B	1000	5253	5253	2.06	t.lim.	5253	—	0.37	5253	—	0.35
n40C	1000	4684	4684	2.60	t.lim.	4684	—	1.40	4684	—	1.78
n40D	1000	5295	5295	—	2772.09	5295	—	0.17	5295	—	0.13
n40E	1000	5209	5209	—	233.20	5209	—	1.23	5209	—	1.31
n40F	1000	4944	4944	—	1003.03	4944	—	5.06	4944	—	4.48
n40G	1000	5312	5312	—	11.78	5312	—	0.16	5312	—	0.17
n40H	1000	4939	4939	1.86	t.lim.	4939	—	13.51	4939	—	12.73
n40I	1000	4912	4912	0.75	t.lim.	4912	—	3.03	4912	—	3.75
n40J	1000	4811	4811	—	9.15	4811	—	0.12	4811	—	0.21

Table 3.8: Results for instances with 50 stations and $q = \{10, 15, 20, 25, 30\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n50A	10	6598	6598	1.36	t.lim.	6598	4.73	t.lim.	6617	5.75	t.lim.
n50B	10	9337	9264	15.73	t.lim.	9054	22.26	t.lim.	9056	1.03	t.lim.
n50C	10	8367	8298	10.25	t.lim.	8364	7.72	t.lim.	8255	2.35	t.lim.
n50D	10	10086	10086	12.37	t.lim.	10028	5.91	t.lim.	—	1.88	t.lim.
n50E	10	9542	9500	18.54	t.lim.	9542	7.27	t.lim.	—	1.10	t.lim.
n50F	10	8172	8172	22.50	t.lim.	8136	12.81	t.lim.	7763	1.89	t.lim.
n50G	10	6986	6986	3.54	t.lim.	6986	—	1160.81	6986	—	235.09
n50H	10	8876	8831	20.40	t.lim.	8876	10.11	t.lim.	9099	10.64	t.lim.
n50I	10	8424	8424	21.49	t.lim.	8275	12.54	t.lim.	8259	6.01	t.lim.
n50J	10	8216	8216	15.77	t.lim.	8216	9.93	t.lim.	8095	2.64	t.lim.
n50A	15	6081	6081	—	788.18	6081	—	20.55	6081	—	35.64
n50B	15	7764	7764	10.48	t.lim.	7764	6.95	t.lim.	—	3.56	t.lim.
n50C	15	6699	6699	2.12	t.lim.	6699	0.46	t.lim.	—	50.66	t.lim.
n50D	15	8199	8199	7.65	t.lim.	8199	3.95	t.lim.	—	2.64	t.lim.
n50E	15	7771	7771	10.58	t.lim.	7771	6.64	t.lim.	7871	4.20	t.lim.
n50F	15	6546	6519	10.58	t.lim.	6519	—	2798.52	6519	—	328.20
n50G	15	6273	6273	—	123.01	6273	—	81.85	6273	—	6.35
n50H	15	7034	7034	5.24	t.lim.	7034	0.99	t.lim.	7005	1.16	t.lim.
n50I	15	6423	6423	1.67	t.lim.	6423	—	296.12	6423	—	49.05
n50J	15	6739	6739	3.92	t.lim.	6699	3.57	t.lim.	—	5.58	t.lim.
n50A	20	5856	5856	—	3387.75	5856	—	2.39	5856	—	1.16
n50B	20	6929	6929	5.26	t.lim.	6929	2.00	t.lim.	6929	4.86	t.lim.
n50C	20	6185	6165	—	761.52	6165	—	110.01	6165	—	44.54
n50D	20	7198	7188	1.28	t.lim.	7198	2.21	t.lim.	7188	—	1117.08
n50E	20	6951	6951	4.13	t.lim.	6951	6.22	t.lim.	6938	1.54	t.lim.
n50F	20	6026	6026	3.76	t.lim.	6026	1.14	t.lim.	—	3.07	t.lim.
n50G	20	5950	5950	—	2664.96	5950	—	0.63	5950	—	1.98
n50H	20	6366	6359	—	2090.95	6359	—	308.06	6359	—	327.19
n50I	20	6041	6041	2.88	t.lim.	6041	—	66.47	6041	—	62.75
n50J	20	6063	6063	—	167.86	6063	—	16.66	6063	—	29.32
n50A	25	5739	5739	—	11.52	5739	—	0.92	5739	—	1.80
n50B	25	6687	6687	3.75	t.lim.	6687	—	1193.82	6687	—	300.31
n50C	25	5828	5820	0.35	t.lim.	5820	—	107.41	5820	—	330.35
n50D	25	6817	6817	1.22	t.lim.	6817	—	202.27	6817	—	69.09
n50E	25	6408	6408	0.56	t.lim.	6408	—	57.25	6408	—	82.86
n50F	25	5804	5789	2.34	t.lim.	5789	—	79.30	5789	—	49.40
n50G	25	5913	5913	0.53	t.lim.	5913	—	0.92	5913	—	1.67
n50H	25	5942	5942	—	37.94	5942	—	4.66	5942	—	5.48
n50I	25	5651	5651	0.71	t.lim.	5651	—	1.35	5651	—	1.25
n50J	25	5861	5861	—	109.87	5861	—	13.07	5861	—	16.92
n50A	30	5688	5688	—	18.27	5688	—	0.38	5688	—	0.49
n50B	30	6420	6420	2.89	t.lim.	6418	—	1157.98	6420	0.49	t.lim.
n50C	30	5681	5681	0.49	t.lim.	5681	—	11.14	5681	—	4.69
n50D	30	6589	6589	2.13	t.lim.	6543	—	109.65	6543	—	489.01
n50E	30	6219	6219	—	41.83	6219	—	2.34	6219	—	4.44
n50F	30	5576	5576	2.17	t.lim.	5576	—	24.46	5576	—	21.27
n50G	30	5864	5864	—	316.91	5864	—	0.28	5864	—	0.47
n50H	30	5804	5804	—	19.25	5804	—	0.95	5804	—	0.98
n50I	30	5534	5534	0.95	t.lim.	5534	—	0.28	5534	—	0.54
n50J	30	5747	5747	—	19.46	5747	—	1.19	5747	—	3.68

Table 3.9: Results for instances with 50 stations and $q = \{35, 40, 45, 1000\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n50A	35	5645	5645	—	5.93	5645	—	0.17	5645	—	0.27
n50B	35	6269	6269	2.03	t.lim.	6269	—	65.35	6269	—	53.97
n50C	35	5595	5595	1.40	t.lim.	5595	—	14.30	5595	—	30.59
n50D	35	6371	6368	0.80	t.lim.	6360	—	13.51	6360	—	34.04
n50E	35	6168	6168	—	44.79	6168	—	1.39	6168	—	1.66
n50F	35	5482	5482	4.20	t.lim.	5482	—	42.76	5482	—	79.75
n50G	35	5864	5864	—	714.45	5864	—	0.29	5864	—	0.42
n50H	35	5708	5708	—	12.33	5708	—	0.41	5708	—	1.28
n50I	35	5534	5534	0.99	t.lim.	5534	—	0.30	5534	—	0.57
n50J	35	5684	5684	—	8.61	5684	—	0.88	5684	—	1.19
n50A	40	5645	5645	—	4.08	5645	—	0.18	5645	—	0.27
n50B	40	6127	6127	1.12	t.lim.	6127	—	20.23	6127	—	23.39
n50C	40	5439	5439	—	22.60	5439	—	0.54	5439	—	0.78
n50D	40	6258	6258	—	27.60	6258	—	1.42	6258	—	1.48
n50E	40	6089	6089	—	20.94	6089	—	0.46	6089	—	0.89
n50F	40	5364	5364	2.63	t.lim.	5364	—	8.88	5364	—	17.11
n50G	40	5864	5864	—	390.88	5864	—	0.30	5864	—	0.43
n50H	40	5708	5708	1.71	t.lim.	5708	—	0.73	5708	—	0.88
n50I	40	5516	5516	0.78	t.lim.	5516	—	0.34	5516	—	0.49
n50J	40	5676	5676	—	13.46	5676	—	0.95	5676	—	1.20
n50A	45	5645	5645	—	6.75	5645	—	0.18	5645	—	0.26
n50B	45	6061	6061	2.86	t.lim.	6061	—	22.48	6061	—	16.21
n50C	45	5337	5337	—	261.00	5337	—	1.96	5337	—	3.21
n50D	45	6209	6209	—	73.86	6209	—	3.71	6209	—	5.04
n50E	45	6089	6089	—	24.43	6089	—	0.40	6089	—	0.58
n50F	45	5364	5364	2.83	t.lim.	5364	—	8.62	5364	—	14.27
n50G	45	5864	5864	—	548.20	5864	—	0.30	5864	—	0.41
n50H	45	5708	5708	1.90	t.lim.	5708	—	0.75	5708	—	0.59
n50I	45	5516	5516	0.78	t.lim.	5516	—	0.34	5516	—	0.46
n50J	45	5676	5676	—	12.38	5676	—	0.98	5676	—	0.96
n50A	1000	5645	5645	—	5.31	5645	—	0.18	5645	—	0.28
n50B	1000	5955	5955	1.11	t.lim.	5955	—	1.94	5955	—	4.69
n50C	1000	5284	5284	2.49	t.lim.	5284	—	8.36	5284	—	11.07
n50D	1000	6111	6111	4.17	t.lim.	6111	—	122.26	6111	—	127.79
n50E	1000	6023	6023	—	153.10	6023	—	0.16	6023	—	0.26
n50F	1000	5349	5349	2.56	t.lim.	5349	—	13.22	5349	—	12.91
n50G	1000	5864	5864	—	604.66	5864	—	0.30	5864	—	0.41
n50H	1000	5708	5708	2.73	t.lim.	5708	—	2.62	5708	—	3.94
n50I	1000	5516	5516	0.79	t.lim.	5516	—	0.34	5516	—	0.46
n50J	1000	5676	5676	—	11.47	5676	—	0.95	5676	—	0.97

Table 3.10: Results for instances with 60 stations and $q = \{10, 15, 20, 25, 30\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n60A	10	8181	8101	16.29	t.lim.	8103	13.99	t.lim.	—	—	t.lim.
n60B	10	8418	8418	20.11	t.lim.	8418	14.36	t.lim.	—	—	t.lim.
n60C	10	9412	9387	16.97	t.lim.	9400	16.01	t.lim.	—	—	t.lim.
n60D	10	10818	10818	23.19	t.lim.	10818	18.98	t.lim.	14648	37.85	t.lim.
n60E	10	9421	9388	19.72	t.lim.	9340	6.63	t.lim.	—	—	t.lim.
n60F	10	8834	8834	23.39	t.lim.	8834	19.43	t.lim.	8987	17.14	t.lim.
n60G	10	8913	8911	17.98	t.lim.	8913	13.99	t.lim.	—	—	t.lim.
n60H	10	8301	8269	21.47	t.lim.	8114	0.28	t.lim.	8426	5.77	t.lim.
n60I	10	9557	9557	20.39	t.lim.	9557	17.94	t.lim.	—	—	t.lim.
n60J	10	8644	8644	21.69	t.lim.	8644	15.56	t.lim.	8397	8.21	t.lim.
n60A	15	6774	6774	8.06	t.lim.	6774	4.07	t.lim.	—	—	t.lim.
n60B	15	6993	6993	6.81	t.lim.	6993	—	1190.60	6993	—	696.12
n60C	15	7724	7708	10.55	t.lim.	7724	7.49	t.lim.	7720	5.23	t.lim.
n60D	15	8470	8470	16.23	t.lim.	8470	12.08	t.lim.	—	—	t.lim.
n60E	15	7569	7543	3.17	t.lim.	7518	—	1690.62	7518	—	637.73
n60F	15	7075	7075	12.78	t.lim.	0	—	0.00	7075	0.93	t.lim.
n60G	15	7364	7364	8.29	t.lim.	7364	4.02	t.lim.	—	—	t.lim.
n60H	15	6829	6829	5.91	t.lim.	6829	1.68	t.lim.	6793	—	1108.06
n60I	15	7660	7660	14.60	t.lim.	7660	10.28	t.lim.	—	—	t.lim.
n60J	15	7043	7043	2.47	t.lim.	7043	0.88	t.lim.	7043	—	201.70
n60A	20	6298	6298	1.65	t.lim.	6298	—	240.90	6298	—	91.09
n60B	20	6448	6448	1.14	t.lim.	6448	—	103.62	6448	—	48.60
n60C	20	7010	7010	3.24	t.lim.	7010	0.92	t.lim.	7010	—	2807.60
n60D	20	7605	7605	11.87	t.lim.	7605	8.48	t.lim.	7600	1.56	t.lim.
n60E	20	7032	7032	0.85	t.lim.	7031	2.41	t.lim.	7031	3.69	t.lim.
n60F	20	6339	6339	0.66	t.lim.	6339	—	72.07	6339	—	18.27
n60G	20	6802	6802	1.14	t.lim.	6802	—	1288.94	6802	—	260.25
n60H	20	6313	6313	0.81	t.lim.	6313	—	187.30	6313	—	46.43
n60I	20	6959	6959	10.39	t.lim.	6959	7.35	t.lim.	—	—	t.lim.
n60J	20	6727	6727	0.95	t.lim.	6727	—	72.27	6727	—	30.96
n60A	25	6147	6147	0.37	t.lim.	6147	—	27.65	6147	—	77.34
n60B	25	6291	6291	—	50.57	6291	—	8.88	6291	—	13.15
n60C	25	6631	6631	—	1314.00	6631	—	45.96	6631	—	82.08
n60D	25	7067	7067	4.73	t.lim.	7067	—	3309.46	7067	—	507.61
n60E	25	6633	6633	—	125.57	6633	—	4.68	6633	—	8.51
n60F	25	6163	6163	2.50	t.lim.	6163	—	88.46	6163	—	187.67
n60G	25	6447	6447	—	508.30	6447	—	6.73	6447	—	6.65
n60H	25	6066	6066	—	259.62	6066	—	205.72	6066	—	160.68
n60I	25	6623	6623	5.80	3402.79	6623	2.81	t.lim.	6623	0.18	t.lim.
n60J	25	6623	6623	1.50	t.lim.	6623	—	28.35	6623	—	65.12
n60A	30	5943	5943	—	36.79	5943	—	1.16	5943	—	2.37
n60B	30	6249	6249	—	1263.94	6249	—	4.33	6249	—	2.37
n60C	30	6394	6394	—	51.86	6394	—	4.76	6394	—	3.19
n60D	30	6642	6642	2.11	t.lim.	6642	—	215.39	6642	—	178.80
n60E	30	6543	6543	—	65.43	6543	—	3.13	6543	—	8.80
n60F	30	6122	6122	2.98	t.lim.	6122	—	43.75	6122	—	14.21
n60G	30	6447	6447	1.05	t.lim.	6447	—	11.19	6447	—	6.90
n60H	30	6058	6058	0.80	t.lim.	6058	—	115.16	6058	—	95.21
n60I	30	6504	6504	2.99	t.lim.	6504	4.44	t.lim.	6576	3.13	t.lim.
n60J	30	6417	6417	—	20.00	6417	—	0.58	6417	—	1.17

Table 3.11: Results for instances with 60 stations and $q = \{35, 40, 45, 1000\}$

		BinArc + initial UB				MEN + initial UB			MEN		
instance	Q	init	ub	gap	sec	ub	gap	sec	ub	gap	sec
n60A	35	5857	5857	—	1293.50	5857	—	1.25	5857	—	0.70
n60B	35	6218	6218	—	115.06	6218	—	2.74	6218	—	1.90
n60C	35	6180	6180	—	10.41	6180	—	0.50	6180	—	0.45
n60D	35	6589	6589	2.11	t.lim.	6589	—	163.12	6589	—	231.99
n60E	35	6357	6357	—	39.29	6357	—	2.38	6357	—	2.09
n60F	35	6085	6085	2.78	t.lim.	6085	—	17.83	6085	—	9.85
n60G	35	6388	6388	1.15	t.lim.	6388	—	3.25	6388	—	4.02
n60H	35	6038	6038	1.83	t.lim.	6011	—	27.51	6011	—	30.51
n60I	35	6314	6314	1.35	t.lim.	6314	2.13	t.lim.	6314	—	1954.71
n60J	35	6374	6374	—	15.03	6374	—	0.30	6374	—	0.56
n60A	40	5857	5857	—	2949.84	5857	—	0.74	5857	—	0.79
n60B	40	6198	6198	—	105.82	6198	—	5.33	6198	—	7.05
n60C	40	6180	6180	—	1101.37	6180	—	6.60	6180	—	10.79
n60D	40	6485	6485	1.41	t.lim.	6485	—	53.35	6485	—	74.45
n60E	40	6254	6254	0.30	t.lim.	6254	—	8.40	6254	—	9.70
n60F	40	5993	5993	2.22	t.lim.	5993	—	4.57	5993	—	10.16
n60G	40	6384	6384	2.07	t.lim.	6384	—	6.37	6384	—	4.99
n60H	40	5992	5992	1.88	t.lim.	5992	—	29.55	5992	—	30.24
n60I	40	6106	6106	0.63	t.lim.	6106	—	106.47	6106	—	81.72
n60J	40	6374	6374	—	12.83	6374	—	0.30	6374	—	0.56
n60A	45	5857	5857	1.04	t.lim.	5857	—	0.74	5857	—	0.81
n60B	45	6198	6198	—	91.16	6198	—	6.88	6198	—	6.47
n60C	45	6180	6180	0.36	t.lim.	6180	—	8.25	6180	—	12.17
n60D	45	6411	6411	2.28	t.lim.	6411	—	112.75	6411	—	109.75
n60E	45	6173	6173	—	8.45	6173	—	0.58	6173	—	0.75
n60F	45	5977	5977	2.88	t.lim.	5977	—	10.23	5977	—	10.16
n60G	45	6375	6375	1.93	t.lim.	6375	—	6.86	6375	—	3.39
n60H	45	5992	5992	2.20	t.lim.	5992	—	73.07	5992	—	74.81
n60I	45	6100	6100	2.31	t.lim.	6100	—	518.61	6100	—	451.40
n60J	45	6374	6374	—	6.29	6374	—	0.31	6374	—	0.57
n60A	1000	5857	5857	1.27	t.lim.	5857	—	0.74	5857	—	0.42
n60B	1000	6198	6198	—	102.07	6198	—	6.42	6198	—	8.37
n60C	1000	6180	6180	0.62	t.lim.	6180	—	17.98	6180	—	14.20
n60D	1000	6297	6297	0.55	t.lim.	6297	—	3.04	6297	—	5.75
n60E	1000	6173	6173	—	8.16	6173	—	0.58	6173	—	0.72
n60F	1000	5944	5944	2.47	t.lim.	5944	—	3.55	5944	—	10.27
n60G	1000	6309	6309	1.92	t.lim.	6309	—	1.98	6309	—	1.07
n60H	1000	5992	5992	2.21	t.lim.	5992	—	116.56	5992	—	200.90
n60I	1000	6046	6046	1.89	t.lim.	6046	—	113.65	6046	—	146.84
n60J	1000	6374	6374	—	5.63	6374	—	0.29	6374	—	0.57

3.8 Bibliography

C. Archetti, M.W.P. Savelsbergh, and M.G. Speranza. Worst-Case Analysis for Split Delivery Vehicle Routing Problems. *Transportation Science*, 40(2):226–234, 2006.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: aclassification scheme and survey. *Top*, 15(1):1–31, 2007.

D. Chemla, F. Meunier, and R. Calvo, Wolfer Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.

C. Contardo, C. Morency, and L.-M. Rousseau. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, Universitè de Montrèal, Montrèal, Canada, 2012.

F. Cruz, B.P. Bruck, A. Subramanian, and M. Iori. A heuristic algorithm for the static bike sharing rebalancing problem. Technical report, Universidade Federal da Paraíba and Università degli Studi di Modena e Reggio Emilia, 2016.

M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega (United Kingdom)*, 45(0):7–19, 2014.

M. Dell'Amico, M. Iori, S. Novellani, and T. Stützle. A Destroy and Repair Algorithm for the Bike sharing Rebalancing Problem. *Computers & Operations Research*, 2016.

Paul DeMaio. Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation*, 12(4):3, 2009.

G. Erdoğan, G. Laporte, and R. Wolfler Calvo. The static bicycle relocation problem with demand intervals. *European Journal of Operational Research*, 238(2):451–457, 2014.

G. Erdoğan, M. Battarra, and R. Wolfler Calvo. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, 245 (3):667–679, 2015.

M. Fischetti and M. Monaci. Exploiting Erraticism in Search. *Operations Research*, 62(1): 114–122, 2014.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1 SPEC. ISS.):126–139, 2004a.

H. Hernández-Pérez and J.-J. Salazar-González. Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. *Transportation Science*, 38(2):245–255, 2004b.

H. Hernández-Pérez and J.-J. Salazar-González. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks*, 50(4):258–272, 2007.

H. Hernández-Pérez, I. Rodríguez-Martín, and J.-J. Salazar-González. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers and Operations Research*, 36(5):1639–1645, 2009.

A. Lodi and A. Tramontani. Performance Variability in Mixed-Integer Programming. *Tutorials INFORMS*, pages 1–12, 2013.

N. Mladenović, D. Urošević, S. Hanafi, and A. Ilić. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research*, 220(1):270–285, jul 2012.

M. Nowak, O. Ergun, and C.C. White. Pickup and Delivery with Split Loads. *Transportation Science*, 42(1):32–43, 2008.

OBIS project. Optimising bike sharing in european cities - a handbook, 2011.

J.-J. Salazar-González and B. Santos-Hernández. The split-demand one-commodity pickup-and-delivery travelling salesman problem. *Transportation Research Part B: Methodological*, 75:58–73, may 2015.

F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86(3):565–594, dec 1999.

F. Wang, A. Lim, and Z. Xu. The one-commodity pickup and delivery travelling salesman problem on a path or a tree. *Networks*, 48(1):24–35, aug 2006.

Chapter 4

Minimizing CO_2 emissions in a practical daily carpooling problem

The increasing awareness on the damages to the environment caused by human activity is getting increased attention from governments, as well as companies and individuals. In this sense, the reduction of CO_2 emissions is an important subject and there are a range of practices that help achieving this goal. An example is carpooling, which is defined as the act of individuals sharing a single car. In this chapter we approach a practical case found in a large Italian company and propose two mathematical formulations and two heuristic algorithms. The objective is to develop an integrated web application to be used by the employees of this company in order to organize carpoolings. Experimental results attest for a great potential in CO_2 savings by the use of carpooling in this scenario.

4.1 Introduction

The debate on the negative effects of human activities on the environment has been around for a long time, and nowadays the concern about these issues is widespread. Pollution is a plague in urban areas and a cause of illness for inhabitants. Large companies daily require that thousands of individuals travel to work, heavily increasing car use. This practice causes a wide range of problems, including high emissions of CO_2 in the atmosphere, noise pollution, and parking issues (see, e.g., Gärling and Friman 2015). Public transportation systems could mitigate the problem, but, more often than not, they are incapable of serving all the demands in a cost-effective manner, or are simply disregarded by many individuals, who prefer not to use them.

In this sense, *carpooling* can be an effective tool to help reduce traffic and pollution. Carpooling is a ridesharing practice that can be defined as the act of a group of individuals that ride a single car by splitting travel costs (see, e.g., Furuhata et al. 2013). This practice has grown more common in recent years. It is an interesting transportation habit for individuals as

well as for companies, as it can reduce transportation costs and directly affect CO_2 emissions. It thus fits well with the millennium goal of the United Nations (2015) that aims at ensuring environmental sustainability. Governments are also taking actions to support the carpooling practice. For instance, in Italy a law for sustainable mobility was included into the national legislation in 1998 to stimulate the use of collective transportation methods and promote the creation of innovative transport systems (see Italian Ministry of Environment 1998). Consequently, in recent years several optimization methods have been developed to provide good solutions to different carpooling practices, both focusing on the solution of real-world study cases (see, e.g., Wolfler Calvo et al. 2004) and on the development of general solution algorithms (see, e.g., Baldacci et al. 2004).

Carpooling is sometimes also denoted as *ridesharing*, although this terminology is more common in dynamic contexts (see, e.g., Agatz et al. 2011, 2012). It should not be confused instead with *carsharing*, which is a kind of decentralized car rental service where travelers are allowed to pick up a car at a station, use it for the time needed, and then return it (to the same station or to a different one, according to the implemented system). Carsharing has also attracted research aimed at the minimization of CO_2 emissions. Lee et al. (2014) pursued indeed this objective by determining the optimal carsharing service locations in Daejeon, a small Korean city. The reader interested in the research conducted on carsharing is referred to the recent survey by Jorge and Correia (2013).

In this chapter we study a practical case found in a large service company, Coopservice Soc.coop.p.A., with headquarters in Reggio Emilia (Italy), whose aim is to encourage its employees to practice carpooling to reduce transportation costs and CO_2 emissions. We focus on a pilot case in which all employees are headed to the same workplace but may have different working shifts. According to the classification by Wolfler Calvo et al. (2004) our case is a *daily* carpooling problem, because users offer or require passages on the basis of working shifts that change day by day. Our intention is to assess, by the use of optimization algorithms and mathematical models, the amount of CO_2 emissions that can be saved in this pilot case, and then develop a web application in which employees can communicate and agree with each other on sharing rides.

The remainder of the chapter is organized as follows. In Section 4.2 we present a brief review of the carpooling literature. In Section 4.3 we formally describe the problem and discuss the details of our case study. In Section 4.4 mathematical models and heuristic algorithms are presented and computationally evaluated. In Section 4.5 a prototype of the web application is shown and then some conclusions are drawn in Section 4.6.

4.2 Brief Review of Carpooling Literature

The practice of carpooling (also known as ridesharing) has attracted the interest of many researchers from different areas, consequently leading to a huge literature. In this section we present a brief review of some interesting results in the field. The reader interested in a deeper review is referred to the following papers: Furuhata et al. (2013), who present a framework to identify key challenges in ridesharing and foster the development of effective formal ridesharing mechanisms; Chan and Shaheen (2012), that focus on how the use of Internet, mobile phones, and social networking has been integrated in carpooling programmes; Battarra et al. (2014) and Döerner and Salazar-González (2014), that survey pickup and delivery problems for, respectively, freight and passengers transportation.

The carpooling literature can be roughly divided into two major approaches: a *deductive* approach and an *inductive* one. The former approach looks at carpooling through the lenses of existing literature from different research areas, and thus tries to apply existing theories (deriving from fields such as behavioral theories and applied psychology) to draw conclusions on the usage and effectiveness of possible practical carpooling systems. From this perspective, deductive studies share a somehow top-down approach. The latter approach examines instead existing carpooling initiatives in order to understand what correlations among variables arise in real contexts, or what individuals think that might lead them to carpool. In this sense, inductive studies start from reality and try to draw general conclusions about the carpooling phenomenon, thus sharing a bottom-up approach.

The main results on deductive and inductive approaches are surveyed in Sections 4.2.1 and 4.2.2, respectively, whereas in Section 4.2.3 we describe the main models and algorithms that have been developed to provide optimized solutions to carpooling problems.

4.2.1 The Deductive Approach

The deductive approach roughly follows the call of Gärling and Schuitema (2007) for a behavioral framework able to help policy makers and carpooling system designers to develop effective car use reduction programs, as well as to support individuals to develop their own car use reduction goals. Gardner and Abraham (2008) proposed an analysis of the literature on psychological correlates of car use with data taken from fourteen previous studies. Their research aim was to verify the impact of the *theory of planned behavior* (TPB) by Ajzen (1991) on car reduction behavior. TPB is an expectancy-value theory in which the subjective perception of the probability of happening of a given behavior directly influences the process of creation of attitudes towards that behavior. Following Gardner and Abraham (2008), Abrahamse et al. (2009) measured the impact of TPB and of the *norm-activation theory* (NAT) by Schwarz (1977) on sample data from Canadian workers. While TPB looks at the mechanism through which behavioral intentions are formed, NAT is more about moral issues,

i.e., about how problem awareness and perceived responsibility for consequences may modify individual attitudes towards a certain behavior. The authors highlight that, individuals' attitudes and the perceived possibilities and difficulties related to reducing car use strongly influence the transportation choice.

Bamberg et al. (2011) also considered TPB and NAT. They proposed and tested a conjoint self-regulation theory asserting that changes in behavior are a process of transition through successive stages. In particular, in the case of a car use reduction goal, the theory starts from the formation of the goal, then it passes through the formation of the behavioral intention to do it, and lastly it comes to choosing the alternative travel option that reduces car use. Shewmake (2012) reviewed studies on the impacts of *high occupancy vehicle* lanes on carpooling, with a focus on behavioral models. Different performance measures related to welfare, congestion, and air pollution effects were surveyed, by taking into consideration papers that explicitly model carpool formation. In conclusion, behavioral theories seem to indicate that, in order to foster mixed transportation methods, both personal interest variables (e.g., the perception of alternatives), as well as personal norms should be taken into account when proposing carpooling systems.

Other important contributions derive from proposals of new models for web or app-based carpooling systems. These models try to devise carpooling systems by taking into account both contributions from the literature and from practical experiences, considering different perspectives such as social, technical, and business sustainability. Selker and Saphir (2010) proposed a carpooling system that takes into account individual interests, goals, and preferences. They argued that having the same interests can reduce some negative psychological drawbacks of carpooling, as, for instance, privacy issues. They also conducted a promising pilot test within a university setting. Chen and Hsu (2013) extended the model in Selker and Saphir (2010), by defining other options aimed at improving the carpooling application. Ribeiro et al. (2015) developed a web and app-based carpooling system, whose design benefits from an analysis of existing carpooling systems. Other web applications, more focused on optimization aspects, are discussed in Section 4.2.3.

4.2.2 The Inductive Approach

Governments and private organizations created several travel plans to reduce environmental costs. This provided researchers with the opportunity to go into the field and study existing applications. In what we call inductive studies, the authors analyzed existing carpooling initiatives and assessed whether and why individuals decided to participate.

In the United Kingdom, Cairns et al. (2010) analyzed the efficacy of the travel plans of twenty employers. Their findings show that the presence of travel plans, in conjunction with parking management techniques, is capable of reducing car use. They also show that this practice can offer positive economic returns for the proposing companies because of the

reduction in car parking requirements.

In Belgium, Vanoutrive et al. (2012) studied commuters' choices by clustering individuals by company. This strategy is grounded on three arguments: employers can foster internal commutation patterns among employees; neighbor companies can have different accessibility options; companies are social settings where organizational culture and social norms can exert a great influence on the individual choice to commute. The authors found that carpooling initiatives could be effective only where public transportation is either weak or absent, which is consistent with previous findings by Teal (1987). They also reported that the carpooling practice had its best results in those sectors where individuals must travel to working sites that change during time, as, e.g., in the construction sector.

In Canada, Buliung et al. (2009) studied *CarpoolZone*, a free-to-use online service that proposes ride matches to commuters living and/or working near each other and having similar travel times. From an analysis conducted on real data, it emerged that individuals living within one kilometer of a carpool lot have greater chances of successfully creating a carpool crew. Moreover, individuals with higher auto availability feel a stronger urge to carpool, perhaps in order to share the costs associated with car travel. In a subsequent study, Buliung et al. (2012) extend the analysis in Buliung et al. (2009) by using data from employers. Their study showed that the typical commuter is relatively young (less than 40) and well-paid, he/she works non-standard hours within small to medium sized companies, and he/she has access to few household vehicles. Females appeared to carpool more frequently than males. In addition, company travel plans proved to be the strongest indicator of carpool formation, especially with designated carpool parking places and emergency ride home services.

Abrahamse and Keall (2012) conducted a study regarding *Let's Carpool*, a New Zealander initiative similar to CarpoolZone but aimed at increasing vehicle occupation. This service is based on a website with an online ride matching algorithm that enables users to organize carpoolings to and from work. Match search options are many and range from a certain maximum distance from home, to other criteria such as non/smoking or gender. The website shows possible matches on a map with markers indicating trip starting and arriving points and generates a list containing contact details of matches. Users are then free to contact potential matches. The study in Abrahamse and Keall (2012) showed that the aspects of carpooling that make it preferable over public transportation are many, including money and time saving, reliability, and sociability. However, users also indicated some negative aspects of carpooling, such as lack of flexibility, absence of emergency cars, and difficult arrangement of costs.

A few studies concentrated on the assessment of individuals' opinions about carpooling. Nielsen et al. (2015) analyzed, through qualitative interviews and focus groups, the perceptions of individuals from Denmark regarding carpooling. The authors highlighted that respondents never reported the environmental impact as a source of motivation towards car-

pooling. They also suggested that carpooling systems should be integrated with social networks, in order to build on the positive sides of carpooling that individuals perceive. A related study was held in the USA by Lee et al. (2015, forthcoming). Through survey data analysis the authors found out that females and younger men are more inclined towards carpooling. Moreover, commuters from urban areas to country areas are more likely to participate in carpooling. Quite surprisingly, attitudinal variables did not help in explaining the attitude towards carpooling.

4.2.3 Mathematical Models and Optimization Methods

When looking for optimized carpooling solutions, the considered time frame becomes an important problem dimension. In this sense, two types of problems emerged in the related literature: the *daily carpooling problem* (DCPP), where users must agree on how to carpool on a daily basis, and the *long-term carpooling problem* (LCPP), where users form groups and organize shared rides that persist during a certain period of time.

Wolfler Calvo et al. (2004) approached a real-life DCPP and developed an integrated system to organize and manage carpooling groups. Given the necessity for a fast response from the system, they opted to use a heuristic algorithm to provide optimized groups. This algorithm is the core of their optimization module and operates as a two-step procedure. In the first step a graph containing the shortest distance among each pair of users is built by using a modified version of the Dijkstra algorithm that considers traffic congestion. In the second step a problem solution is obtained by executing on the graph a greedy constructive heuristic and a refinement based on local search. Results show that instances with up to 400 employees are handled effectively. Another DCPP was solved by Baldacci et al. (2004), by the use of an exact and a heuristic solution method. The exact method is a bounding based iterative procedure where at each iteration a reduced set-partitioning problem is solved, whereas the heuristic is a Lagrangian constructive procedure. Both approaches were tested on instances derived from the literature and yielded good results.

As for the LCPP, Yan et al. (2011) considered groups of individuals that would like to travel together rather than single individuals. They proposed a multiple commodity network flow formulation, whose objective function considers both the reduction of global costs and a fair sharing of costs among users. Furthermore, they developed a Lagrangian relaxation with subgradient optimization to provide valid lower bounds, and a Lagrangian heuristic to efficiently obtain upper bounds. The resulting solution method was tested on instances from the literature and the results indicated that it could be used to efficiently solve large size instances in a real case.

Naoum-Sawaya et al. (2015) also studied a LCPP derived from a company application. In their problem a fleet of company owned vehicles must be distributed among employees. An employee that is assigned a vehicle is expected to participate in a carpool, bringing other

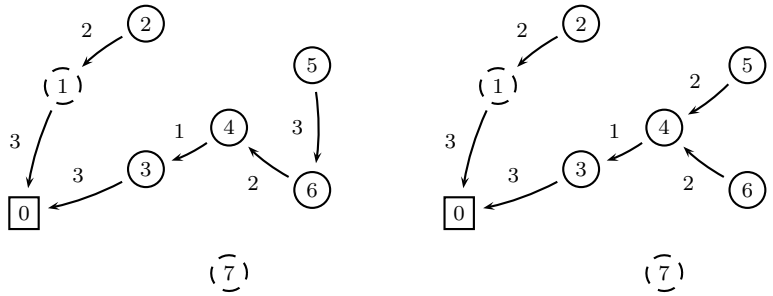
employees to and from work. They proposed a stochastic mixed integer programming model to solve the problem while considering events in which a vehicle may be unavailable, possibly requiring rerouting. They also presented a two-step heuristic in which at first the carpooling crews are created using the well known savings heuristic, and then the stochastic model is used to define which employees should be given the vehicles. Both approaches were tested on several instances of different sizes obtaining a good computational performance.

Instead of focusing on a particular time frame, Xia et al. (2015) distinguished carpooling mainly by how users are matched into carpools and what type of information is used to achieve that. Indeed, some services consider only users requests and do not take into account spacial information to provide more suitable matchings. The authors focused on a carpooling application that makes use of both network and spacial information to match users. They proposed an integer linear programming approach, a tabu search, and a simulated annealing algorithm to solve the problem. The efficiency of their approach was evaluated under different scenarios. Another interesting study focusing on a specific scenario was performed by Bruglieri et al. (2011), who designed a web application to support the use of carpooling in two universities in Milan. The core of their system is built upon a heuristic based on a guided Monte Carlo method and minimizing a multi-objective weighted function that considers distance traveled, level of service, and user preferences. Computational tests indicated that their algorithm was capable of generating good quality solutions.

4.3 Problem Description and Case Study

To formally introduce our DCPP, let us define $G = (V, A)$ as a directed graph, where $V = \{0\} \cup V'$, vertex 0 represents the workplace, vertices in V' are the employees, and $A = \{(i, j) : i \in V', j \in V, i \neq j\}$. For convenience of notation, let us partition the set of employees into two subsets, namely, those that own a car, V_c, and those that do not, V_n. Each employee $i \in V_c$ either uses his/her own car to go directly to the workplace, or carpools by taking a ride from another employee. Each vertex $j \in V_n$ either carpools or uses the public transport to reach the workplace. Vehicles have a maximum capacity of Q people. With every arc $(i, j) \in A$ is associated a non-negative cost c_{ij}, specifying the distance in km between the two vertices. In our problem we consider real distances on an urban area, so the cost matrix c is asymmetric. Let γ_c and γ_b be two coefficients estimating the amount of CO_2 emissions in kg for each km traveled by car and by public transport, respectively. Let also δ be the maximum detour, that is, the maximum allowed distance of an employee path with respect to the shortest path connecting directly the employee to the workplace. In other words, an employee i is not willing to carpool with his/her car for a path that requires more then δc_{io} km. Our DCPP is to drive all employees to the workplace by satisfying the maximum detour constraint, with the aim of minimizing the total CO_2 emission.

We consider two different problem types according to the specific solutions that we allow. In the first type, called *direct-route* DCPP (DCPP_{DR}), a solution is a set of routes such that each of them begins at the house of a given employee $i \in V_c$ and ends at the workplace. In this case, the first employee on the route drives his/her car and picks up the others along the route. In the second type, called *tree-route* DCPP (DCPP_{TR}), we allow employees to drive to intermediary points and then use a single car from there on. These intermediate points can be, in general, parking lots or employees houses. The latter type of solutions allows for some additional flexibility, but it also requires more organization and commitment among participants. This might become an inconvenience in some cases, due to waiting times and lack of parking spots at the meeting points. On the other hand, because of their more restrictive nature, direct-route solutions tend to result in higher costs in terms of both traveled distance and CO_2 emissions. Examples of direct-route and tree-route solutions are shown in Figures 4.1-(a) and 4.1-(b), respectively. The workplace is vertex 0, vertices in solid and dashed lines are employees that own or do not own a car, respectively. The values on the arcs correspond to the c_{ij} distances and $Q = 4$.



(a) *direct-route* solution (b) *tree-route* solution

Figure 4.1: Problem types DCPP_{DR} (left) and DCPP_{TR} (right). Vertex 0 is the workplace, vertices in solid (resp. dashed) lines are employees that own (resp. do not own) a car.

The DCPP_{TR} is more general than the DCPP_{DR}. The DCPP_{DR} is already NP-hard because it contains as a special case the vehicle routing problem with unit demands (see, e.g., Baldacci et al. 2004).

4.3.1 Data Analysis

Coopservice S.coop.P.A. provides services on several sectors, including facility management, security, waste disposal, and third-party logistics. It has a total workforce of about 11 000 people and operates all over the Italian territory. The company provided us with a pilot case containing data for 135 employees working in the hospital Santa Maria Nuova, located in Reggio Emilia (Italy), close to the company headquarters. Most employees perform

cleaning activities at the hospital and have working shifts that change day-by-day.

The data include detailed information on the shifts for a period ranging from February to December 2012. A first analysis that we performed to understand the distribution of the shifts along the week is shown in Figure 4.2. The figure shows the number of employees working per week day. This number is usually around 80 during working days, decreasing to about 70 during Saturdays, and to 40 on Sundays. A direct consequence of the small number of employees working on Sundays is that there are considerably less carpooling options on these days. Therefore, we removed data regarding Sunday shifts from our study, ending up with a set of 276 days. The few evident fluctuations in Figure 4.2 represent holidays with a small number of employees at work. These situations are predictable and do not have a significant impact on the overall performance of the carpooling application.

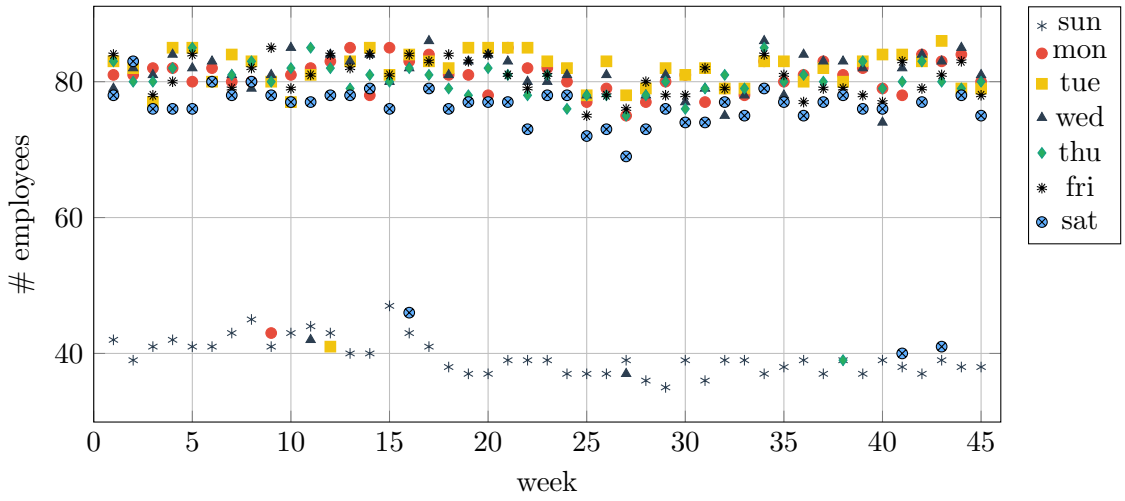


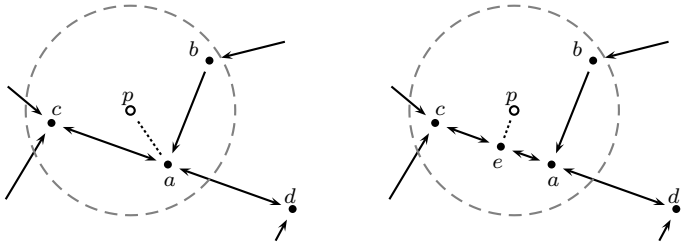
Figure 4.2: Distribution of employees per weekday.

An important factor, that might heavily affect the probability of success of a carpooling application in this context, is the number of different shift typologies. Indeed, the higher is the number of different shifts, the fewer are the possible matching options for carpooling. We found a total of 171 different shift typologies during the period considered on the study. However, some of them are much more common than others. For instance, the shifts 6:00-12:30 and 14:00-20:30 are the most frequent, with frequency rates of 27% and 22%, respectively. Furthermore, the 11 most common shifts account for more than 80% of all entries. These findings seem to suggest that deploying a carpooling application in this scenario could lead to interesting reductions in traveled distance and CO_2 emissions.

To evaluate the distances between each pair of vertices, we developed an application that takes as input a digital map of the region and a list of the geographic coordinates of employee houses and workplace, and produces as output a matrix containing the value of the shortest paths between each pair of points. The application first reads the map and represents it as a

directed graph. Then it takes the list of coordinates for which the distance matrix should be evaluated.

A *map-matching* procedure is necessary to properly insert the set of coordinates into the graph (see, e.g., Quddus et al. 2007). To this aim we implemented a simple two-step procedure that works as follows. Suppose a given point p must be located in the map. At first, we search for all vertices in a given distance range from p and select the *nearest vertex* as the best candidate for the matching. An example is depicted in Figure 4.3-(a), where the nearest vertex is a. In case no vertex is found, we expand the diameter of the range and repeat the process until a match is found. Secondly, we project p into all edges in a given range, and determine the *nearest projection*. In the example shown in Figure 4.3-(b) the nearest projection is e, over the arc (a, c). The option giving the minimum distance is then chosen. If this results in the nearest vertex, then we simple match p into this vertex. If instead it results in the nearest projection, then we divide the selected arc into two arcs, create a new vertex corresponding to the projection, and then map p into the new vertex.



(a) Nearest point in range (b) Nearest edge projection

Figure 4.3: Details of the map-matching procedure.

4.4 Solution Algorithms and Computational Evaluation

In this section we present some solution algorithms that we implemented and then computationally evaluate them on our study case. We first propose a three-index integer linear programming formulation for the case of tree-route solutions. The formulation makes use of three sets of binary variables: x_{ij}^k takes value 1 if employee k travels along arc (i, j), for $(i, j) \in A, k \in V'$; y_{ij} takes value 1 if arc (i, j) is traveled by a vehicle, for $(i, j) \in A$; and v_i takes value 1 if employee i uses public transport instead of carpooling, for $i \in V_n$. The DCPP_{TR} can be modeled as the following *tree-route formulation* (F_{TR}).

$$(\text{F}_{\text{TR}}) \quad \min \; z_{F_{\text{TR}}} = \sum_{(i,j)\in A} \gamma_c c_{ij} y_{ij} + \sum_{i \in V_n} \gamma_b c_{i0} v_i \tag{4.1}$$

subject to

$$\sum_{j \in V} x_{kj}^k = 1 \qquad \forall \; k \in V_c, \tag{4.2}$$

$$\sum_{j \in V} x_{kj}^k = 1 - v_k \qquad \forall\, k \in V_n, \tag{4.3}$$

$$\sum_{i \in V'} x_{i0}^k = 1 \qquad \forall\, k \in V_c, \tag{4.4}$$

$$\sum_{i \in V'} x_{i0}^k = 1 - v_k \qquad \forall\, k \in V_n, \tag{4.5}$$

$$\sum_{i \in V'} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall\, j, k \in V', j \neq k, \tag{4.6}$$

$$\sum_{k \in V'} x_{ij}^k \leq Q y_{ij} \qquad \forall\, (i,j) \in A, \tag{4.7}$$

$$\sum_{i \in V'} \sum_{j \in V} c_{ij} x_{ij}^k \leq c_{k0}(1+\delta) \qquad \forall\, k \in V', \tag{4.8}$$

$$\sum_{j \in V} y_{ij} = 1 \qquad \forall\, i \in V_c, \tag{4.9}$$

$$\sum_{j \in V} y_{ij} = 1 - v_i \qquad \forall\, i \in V_n, \tag{4.10}$$

$$\sum_{i \in V} y_{ij} \geq 1 - v_j \qquad \forall\, j \in V_n, \tag{4.11}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, (i,j) \in A, k \in V', \tag{4.12}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A, \tag{4.13}$$

$$v_i \in \{0,1\} \qquad \forall\, i \in V_n. \tag{4.14}$$

Objective function (4.1) minimizes the total CO_2 emissions. Constraints (4.2) impose that all employees that own a car must be in a route, whereas (4.3) state that employees without a car might go to work by carpooling or by public transport. Constraints (4.4) and (4.5) ensure that all employees that are in a route arrive at the working place, whereas (4.6) impose vehicle flow conservation. Car capacity and maximum detour cannot be exceeded, as stated by (4.7) and (4.8), respectively. Constraints (4.9) and (4.10) define that if an employee carpools, then exactly one car leaves from his/her house, and (4.11) specify that an employee $i \in V_n$ is either picked up by someone or does not carpool. Finally, (4.12), (4.13), and (4.14) define the variables' domains.

Formulation $\mathrm{F_{TR}}$ allows tree-route solutions because the number of arcs entering a given vertex is not restricted. It can be easily adapted to the case of direct-route solutions by introducing the following set of constraints:

$$\sum_{i \in V'} y_{ij} \leq 1 \qquad \forall\, j \in V'. \tag{4.15}$$

Thus the $\mathrm{DCPP_{DR}}$ can be solved by a *direct-route formulation* ($\mathrm{F_{DR}}$) defined by (4.1)–(4.15).

Formulations $\mathrm{F_{TR}}$ and $\mathrm{F_{DR}}$ can be strengthened by fixing arcs that violate the maximum detour. Indeed, we can set $y_{ij} = 0$ for all $i, j \in V' : c_{ij} \geq c_{i0}(1+\delta)$. These formulations are very simple and could be improved in many other ways. However, in this work we use them only with the purpose of providing an estimation of the total CO_2 emission that could

result in our practical application. As both formulations could solve all our instances in small computational times (refer to Section 4.4.1 below), we did not attempt to further improve them, but focused instead on the development of quick heuristics to be embedded into our web application.

The first heuristic, called $\mathrm{H_{DR}}$, solves the $\mathrm{DCPP_{DR}}$ and is guided by a regret measure. It tries to overcome the problem of postponing the insertion of difficult requests by considering a lookahead information. Basically, the measure of the regret of a certain request is calculated by taking the cheapest insertion cost and subtracting it from the second cheapest cost. Requests with a high regret value are probably critical, in the sense that they are not very flexible and, if left unattended for long, could increase consistently the resulting solution value.

Algorithm $\mathrm{H_{DR}}$ starts from the empty solution and then proceeds as follows. It selects the employee $i \in V_c$ who has the highest regret value, and it initializes a new route by connecting i directly to the workplace. At this point the route is simply a direct path to the workplace and i is the driver. The algorithm then proceeds by selecting the next employee that has not yet been routed and has the highest regret value, and inserts it into the best feasible position along the route. When no further feasible insertion is possible, then the current route is closed and a new one is initialized as before. The process is iterated until all employees owning a car have been routed. The remaining employees, if any, are sent to the workplace by public transport and then the total CO_2 emissions are computed.

The second heuristic algorithm that we developed, called $\mathrm{H_{TR}}$, solves the $\mathrm{DCPP_{TR}}$. This algorithm is based on the heuristic proposed by Esau and Williams (1966) for the capacitated minimum spanning tree, and it is guided by the evaluation of savings. Consider the example depicted in Figure 4.4 and, for the sake of simplicity, assume that the route capacity is 5. Let us define the *gate* $g(i)$ of a vertex i as the next vertex in the current route of i to the workplace. Then, the saving of inserting a vertex i in a different route after a vertex j is $s_{ij} = c_{i,g(i)} - c_{ij}$. For instance, in Figure 4.4 $g_5 = 4$ and $s_{52} = 5 - 1 = 4$, so it might be profitable to connect 5 with 2. This choice clearly affects customer 6, whose route is changed, and customer 4, who should become a driver.

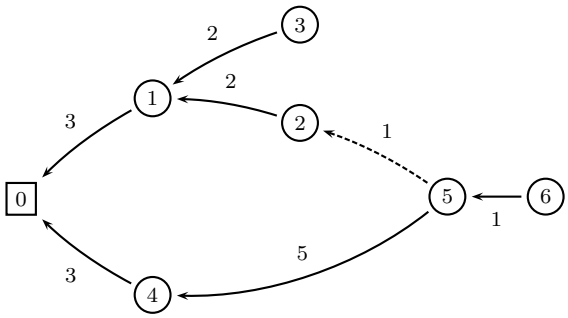


Figure 4.4: Savings evaluation for algorithm $\mathrm{H_{TR}}$.

Algorithm H_{TR} initially builds a solution in which every employee is directly connected to the workplace by a single arc. In this initial solution employees $i \in V_c$ use their car and employees $i \in V_n$ use public transport. Then, the following 4-step procedure is invoked:

Step 1: Find two vertices i and j belonging to different routes and yielding the feasible saving of largest positive value, if any. If no such pair is found, then go to Step 3;

Step 2: Remove arc $(i, g(i))$ from the solution, add arc (i, j), set the new gate of i as $g(i) = j$, and then return to Step 1;

Step 3: Remove from the solution the arc $(i, g(i))$ of any $i \in V_n$ that is assigned to the beginning of a route (as he/she cannot be a driver), and reroute i directly to the workplace by public transport. Repeat until all routes begin with a vertex $i \in V_c$;

Step 4: Evaluate the resulting CO_2 emissions.

4.4.1 Computational Evaluation

In this section we present a computational evaluation of the mathematical formulations and heuristic algorithms that we developed. The coefficients for relating CO_2 emissions to the traveled distances have been set to $\gamma_c = 0.17$ kg/km and $\gamma_b = 0.07$ kg/km, following LCA-lab s.r.l. (2010). The maximum detour from the direct path to the workplace has been set to δ=17%, in accordance with Rietveld et al. (1999). The digital map of the region was taken from the free OpenStreetMap database (`http://www.openstreetmap.org/`). All algorithms were implemented in C++ and the computational experiments were performed on a PC with Intel Core i7-3770 3.40 GHz and 8 Gb of RAM. The *integer linear programs* were solved with IBM Cplex 12.6.1, using the default options. The tests were performed on the 276 working days discussed in Section 4.3.1.

Clearly an employee has to perform two trips every day, one to go to the workplace and another one to return. This allows for some flexibility on how to select which employees can carpool together. Basically, we decided to perform our evaluation on two different scenarios. In the first scenario, called *two-way grouping* (2-way), we attempt to group employees that have the exact same shift, i.e., the same start times for both outbound and return trips. Our algorithms are then invoked to determine the CO_2 emissions for the outbound trip. The emissions for the return trip are evaluated by reverting the (asymmetric) cost matrix by setting $c_{ij} = c_{ji}$ for all $(i, j) \in A$, and then executing again our algorithms. The daily emissions are evaluated by summing the two solution values. In the second scenario, called *one-way grouping* (1-way), we propose a different solution for each of the two trips, attempting to group employees that have the same start time in the outbound trip independently from the return trip, or vice versa. In this case carpooling solutions are computed for both trips, once again taking into account the asymmetric matrix but in this case also considering possible different groups.

In Table 4.1 we present average results per weekday for the 2-way scenario. In column 'no CP' we provide the average daily emissions for the *non-carpooling* case. This corresponds to the solution in which every employee $i \in V_c$ takes his/her car to go directly to the workplace and every employee $i \in V_n$ uses public transport. This clearly provides an *upper bound* on the optimal solution value. For each of our four approaches we then show the average CO_2 emissions in kg and the percentage reductions with respect to the no CP case. We only report average solution times (in seconds) for the formulations, because the heuristic algorithms are practically instantaneous on our benchmark instances.

Exact approaches manage to reduce on average the emissions by around 24.2%, which is a great improvement and shows the potential benefits of carpooling. Also the heuristic algorithm H_{TR} significantly reduces CO_2 emissions, showing a surprisingly great potential for the practical application and yielding high quality solutions that are very close to the optimal ones. The poorer performance of H_{DR} might be due to the rather naive greedy approach of only using the regret measure to create solutions. However, it still yields reductions of about 12.4% on average.

	no CP	H_{DR}		H_{TR}		F_{DR}			F_{TR}		
	CO_2(kg)	CO_2(kg)	red(%)	CO_2(kg)	red(%)	CO_2(kg)	red(%)	sec	CO_2(kg)	red(%)	sec
Mon	94.85	84.38	11.0	74.36	21.6	73.99	22.0	1.07	73.10	22.9	0.99
Tue	92.81	81.60	12.1	72.34	22.1	71.84	22.6	1.13	70.90	23.6	1.09
Wed	93.64	82.19	12.2	72.57	22.5	71.95	23.2	1.23	71.13	24.0	1.11
Thu	95.20	84.00	11.8	74.23	22.0	73.88	22.4	1.00	72.90	23.4	1.01
Fri	96.34	85.02	11.8	74.88	22.3	74.42	22.8	0.98	73.41	23.8	0.90
Sat	90.30	76.53	15.2	66.49	26.4	66.08	26.8	1.01	65.31	27.7	1.00
avg	93.86	82.29	12.4	72.48	22.8	72.03	23.3	1.07	71.13	24.2	1.02

Table 4.1: Reductions over the non carpooling case (no CP) for the 2-way scenario.

Similarly, Table 4.2 shows average results for the 1-way scenario. The gain within this type of scenario is even greater, and formulation F_{TR} manages to reach an average reduction of 30%. The larger reductions obtained with respect to the 2-way scenario are due to the fact that, for the 1-way, the sizes of the groups of employees that can carpool together are larger. The larger group sizes are also the cause of the slightly longer computing times required by the formulations, which however remain very low.

In Figure 4.5 we depict once again the average percentage reductions in CO_2 emissions with respect to the no CP case, but in this case we highlight the evolution for the 45 weeks of the considered time interval. The number of the week is reported on the x-axis, and the average percentage reduction obtained in the week by a given optimization technique on the y-axis. Results by all the developed techniques, considering both the 2-way and 1-way cases, are presented.

The highest possible reductions are obtained for the 1-way case, for which the two formulations yielding tree-route and direct-route solutions show quite similar results. The potential

	no CP	H_{DR}		H_{TR}		F_{DR}			F_{TR}		
	CO_2(kg)	CO_2(kg)	red(%)	CO_2(kg)	red(%)	CO_2(kg)	red(%)	sec	CO_2(kg)	red(%)	sec
Mon	94.85	80.00	15.7	69.01	27.2	68.00	28.3	3.28	67.14	29.2	3.24
Tue	92.81	77.47	16.5	67.07	27.7	65.93	29.0	3.91	65.05	29.9	3.85
Wed	93.64	77.93	16.8	67.67	27.7	66.40	29.1	3.98	65.64	29.9	3.68
Thu	95.20	79.42	16.6	69.01	27.5	67.98	28.6	4.54	67.08	29.5	4.17
Fri	96.34	80.47	16.5	69.36	28.0	68.34	29.1	3.67	67.43	30.0	3.78
Sat	90.30	74.12	17.9	63.50	29.7	62.64	30.6	2.61	61.85	31.5	2.82
avg	93.86	78.23	16.7	67.60	28.0	66.55	29.1	3.66	65.70	30.0	3.59

Table 4.2: Reductions over the non carpooling case (no CP) for the 1-way scenario.

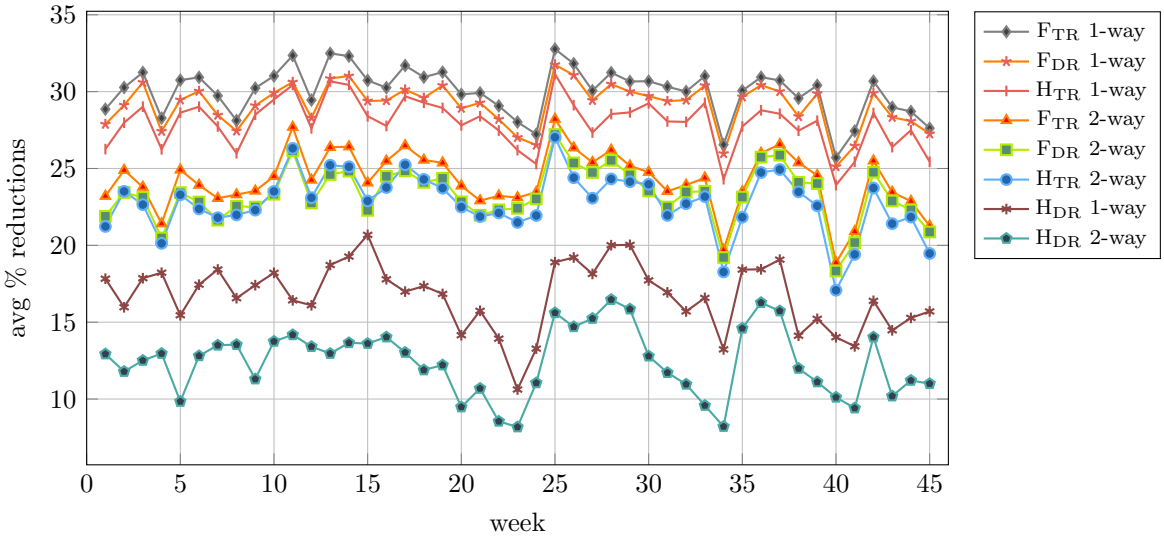


Figure 4.5: Reductions over the non carpooling case per week.

savings found by F_{TR} ranges between 119 to 204 CO_2 kg per week, which corresponds to reductions from 25% to 32%. Also the results by the heuristic based on tree-routes are very good and quite close to those of the two formulations. Overall these three techniques allow to obtain reductions that are usually between 28% and 30% of the emissions of the non-carpooling case. Comprehensibly, the 2-way case shows lower reductions, but also in this case the performance of F_{TR}, F_{DR}, and H_{TR} are very close to one another, usually ranging between 22% and 24%. The performance of H_{DR} is instead considerably lower than that of the above techniques, for both 1-way and 2-way cases. Among the two developed heuristics, H_{TR} should be preferred because it always provides better solutions. By evaluating the total emissions in the whole period of 276 days, we found out that the total difference between the non-carpooling case (worst scenario) and the solution by F_{TR} under the 1-way case (best scenario) amounts to approximately 7.7 tons of CO_2.

4.5 Web Application

A working prototype of a web application has been deployed at the company for testing. The application is divided into two main modules, the *optimization core* and the *visual core*, that communicate with each other by means of a MySQL database to store and retrieve data processed during operations.

Figure 4.6 depicts the basic structure of the application. The optimization core contains two subroutines and, similar to what is done in Wolfler Calvo et al. (2004), it runs on a daily basis to provide carpooling solutions for the following seven days. The *MapAnalyzer* is first run to analyze information regarding the employees of the company and update the file containing the shortest paths, if needed, by accessing the OpenStreetMap database (saved in ShapeFile format). Then, the routine *GenSolutions* reads information regarding the employees and their shortest paths, invokes one of our heuristic algorithms, and stores the solutions in the database using a GeoJSON format. The prototype is equipped with both heuristic algorithms that we developed (so it can solve both DCPP_{DR} and DCPP_{TR}) and uses H_{TR} as default. The server side part (the *back-end* in Figure 4.6) of this module was built using Java EE technology interacting with a MySQL database, while the client side part (the *front-end* in Figure 4.6) was implemented using HTML, JavaServer Pages, and JavaServer Faces.

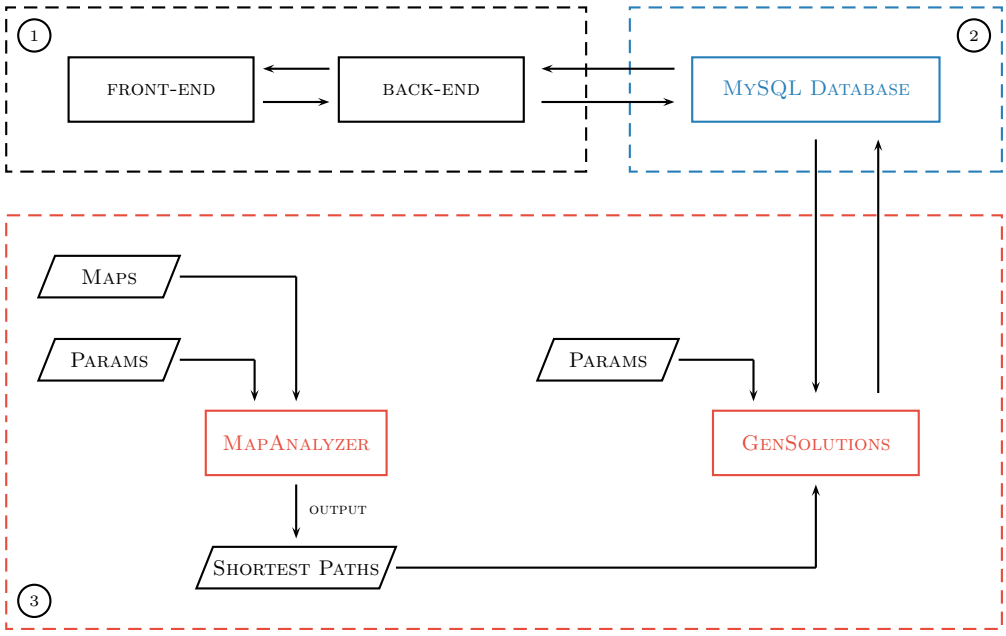


Figure 4.6: System architecture and communication between modules (1- visual core; 2- database; 3- optimization core).

The application is available only to those employees of Coopservice S.coop.P.A. who decided to participate in the carpooling process. Every time a user access the application via the web, the visual core reads from the database the suggestions proposed by the optimization

core and shows the appropriate ones to the user. The user can then communicate via email with other participants, choosing them among the suggested carpools or by simply looking at the closest ones. Figure 4.7 shows a simple screen shot of the application, where a user sees on the map his/her location, the working place, and the shortest path to get there. Notice that the system does not force any carpool and users might refuse to accept a ride request. Moreover it does not suggest a way for sharing costs, as this aspect is left to common agreements among participants.

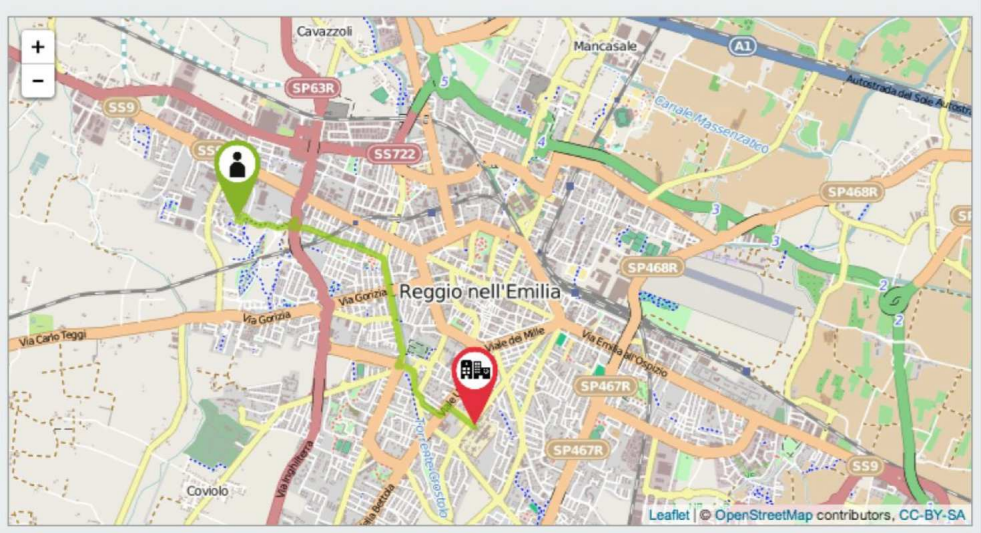


Figure 4.7: Web application prototype: view of the direct route to the workplace.

4.6 Conclusions

In this chapter we approached a practical daily carpooling problem with the aim of reducing CO_2 emissions. We presented a brief survey of the related literature, and then we solved a real-world case study by means of two mathematical formulations and two heuristic algorithms. Moreover, we developed a prototype of a web application. In this application users are provided with the suggested carpools obtained by the use of the heuristics, and can easily communicate with each other to organize carpools according to their wishes. The computational experiments that we performed indicate a great potential for CO_2 reductions. More specifically, under the 2-way grouping strategy (where employees use the same carpool both on the way to work and on the return way) the potential reduction is about 24%, while for the 1-way grouping strategy (where employees can use different carpools for the two trips) this number increases to about 30%. The results also indicate that the company can directly affect the efficiency of the carpooling practice by optimizing employees' shifts in order to create more denser groups, which is an interesting direction for future research. Another interesting direction consists in finding appropriate strategies for sharing costs among users.

4.7 Bibliography

W. Abrahamse and M. Keall. Effectiveness of a web-based intervention to encourage carpooling to work: A case study of Wellington, New Zealand. *Transport Policy*, 21:45–51, 2012.

W. Abrahamse, L. Steg, R. Gifford, and C. Vlek. Factors influencing car use for commuting and the intention to reduce it: A question of self-interest or morality? *Transportation Research Part F: Traffic Psychology and Behaviour*, 12(4):317–324, 2009.

N. Agatz, A. Erera, M. Savelsbergh, and X. Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

N.A.H. Agatz, A.L. Erera, M.W.P. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, 45(9): 1450–1464, 2011.

I. Ajzen. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50(2):179–211, 1991.

R. Baldacci, V. Maniezzo, and A. Mingozzi. An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52(3):422–439, 2004.

S. Bamberg, S. Fujii, M. Friman, and T. Gärling. Behaviour theory and soft transport policy measures. *Transport Policy*, 18(1):228–235, 2011.

M. Battarra, J.-F. Cordeau, and M. Iori. Pickup and delivery problems for goods transportation. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, Monographs on Discrete Mathematics and Applications, pages 161–192. SIAM, 2nd edition, 2014.

M. Bruglieri, D. Ciccarelli, A. Colorni, and A. Lué. PoliUniPool: A carpooling system for universities. *Procedia - Social and Behavioral Sciences*, 20:558–567, 2011.

R. N. Buliung, K. Soltys, C. Habel, and R. Lanyon. Driving factors behind successful carpool formation and use. *Transportation Research Record: Journal of the Transportation Research Board*, 2118(1):31–38, 2009.

R. N. Buliung, R. Bui, and R. Lanyon. When the Internet is not enough: Toward an understanding of carpool services for service workers. *Transportation*, 39(5):877–893, 2012.

S. Cairns, C. Newson, and A. Davis. Understanding successful workplace travel initiatives in the UK. *Transportation Research Part A: Policy and Practice*, 44(7):473–494, 2010.

N.D. Chan and S.A. Shaheen. Ridesharing in North America: Past, present, and future. *Transport Reviews*, 32:93–112, 2012.

Y. T. Chen and C. H. Hsu. Improve the carpooling applications with using a social community based travel cost reduction mechanism. *International Journal of Social Science and Humanity*, 3(2):87–91, 2013.

K. Döerner and J.-J. Salazar-González. Pickup and delivery routing problems for people transportation. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, pages 193–212. SIAM, 2nd edition, 2014.

L.R. Esau and K.C. Williams. On teleprocessing system design. Part 2. *IBM System Journal*, 5(3):142–147, 1966.

M. Furuhata, M. Dessouky, F. Ordóñez, M. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

B. Gardner and C. Abraham. Psychological correlates of car use: A meta-analysis. *Transportation Research Part F: Traffic Psychology and Behaviour*, 11(4):300–311, 2008.

T. Gärling and M. Friman. Unsustainable travel becoming (more) sustainable. In L.A. Reisch and J. Thogersen, editors, *Handbook of Research on Sustainable Consumption*, pages 163–177. Edward Elgar Publishing, 2015.

T. Gärling and G. Schuitema. Travel demand management targeting reduced private car use: Effectiveness, public acceptability and political feasibility. *Journal of Social Issues*, 63(1): 139–153, 2007.

Italian Ministry of Environment. Mobilità sostenibile nelle aree urbane. decreto ministeriale del 27 marzo 1998. Published on the Official Journal of the Italian Republic n. 179, 02/08/1998 (in Italian), 1998.

D. Jorge and G. Correia. Carsharing systems demand estimation and defined operations: A literature review. *European Journal of Transport and Infrastructure Research*, 13:201–220, 2013.

LCA-lab s.r.l. Life cycle assessment applicato alla mobilità del comune di firenze (in italian). Technical report, Bologna, Italy, 2010.

B. H.-Y. Lee, L. Aultman-Hall, M. Coogan, and T. Adler. Rideshare mode potential in non-metropolitan areas of the northeastern united states. *Journal of Transport and Land Use*, 2015, forthcoming.

J.-B. Lee, W. Byun, S.H. Lee, and M. Do. Correlation between optimal carsharing locations and carbon dioxide emissions in urban areas. *International Journal of Environmental Science and Technology*, 11(8):2319–2328, 2014.

J. Naoum-Sawaya, R. Cogill, B. Ghaddar, S. Sajja, R. Shorten, N. Taheri, P. Tommasi, R. Verago, and F. Wirth. Stochastic optimization approach for the car placement problem in ridesharing systems. *Transportation Research Part B: Methodological*, 80:173–184, 2015.

J.R. Nielsen, H. Hovmøller, P. Blyth, and B.K. Sovacool. Of "white crows"and "cash savers:"A qualitative study of travel behavior and perceptions of ridesharing in Denmark. *Transportation Research Part A: Policy and Practice*, 78:113–123, 2015.

M.A. Quddus, W.Y. Ochieng, and R.B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312 – 328, 2007.

Á. Ribeiro, D. C. Silva, and P. H. Abreu. Mocas: Mobile carpooling system. In A. Rocha, A.M. Correia, S. Costanzo, and L.P. Reis, editors, *New Contributions in Information Systems and Technologies*, volume 353 of *Advances in Intelligent Systems and Computing*, pages 913–922. Springer International Publishing, 2015.

P. Rietveld, B. Zwart, B. van Wee, and T. van den Hoorn. On the relationship between travel time and travel distance of commuters. *The Annals of Regional Science*, 33(3):269–287, 1999.

S.H. Schwarz. Normative influences on altruism. *Advances in experimental social psychology*, 10:221–279, 1977.

T. Selker and P.H. Saphir. Travelrole: A carpooling/physical social network creator. In *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*, pages 629–634, Chicago, IL, 2010.

S. Shewmake. Can carpooling clear the road and clean the air? Evidence from the literature on the impact of HOV lanes on VMT and air pollution. *Journal of Planning Literature*, 27 (4):363–374, 2012.

R.F. Teal. Carpooling: Who, how and why. *Transportation Research Part A: General*, 21(3): 203–214, 1987.

United Nations. Millennium development goals, 2015. Available at `http://www.un.org/millenniumgoals/`.

T. Vanoutrive, E. van de Vijver, L. van Malderen, B. Jourquin, I. Thomas, A. Verhetsel, and F. Witlox. What determines carpooling to workplaces in Belgium: Location, organisation, or promotion? *Journal of Transport Geography*, 22:77–86, 2012.

R. Wolfler Calvo, F. de Luigi, P. Haastrup, and V. Maniezzo. A distributed geographic information system for the daily car pooling problem. *Computers & Operations Research*, 31(13):2263–2278, 2004.

J. Xia, K.M. Curtin, W. Li, and Y. Zhao. A new model for a carpool matching service. *PLoS ONE*, 10:e0129257, 2015.

S. Yan, C. Y. Chen, and Y. F. Lin. A model with a heuristic algorithm for solving the long-term many-to-many car pooling problem. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1362–1373, 2011.

Chapter 5

A practical time slot management problem in attended home delivery

In this chapter we approach a practical attended home delivery problem found in a large Italian company, which offers gas and water services to millions of habitants in the region of Emilia Romagna (Italy). According to a series of laws established by the European Union, all companies operating in water and gas distributions had to implement an online application in which they offer certain time slots to fix appointments. Customers can access the application and request services by booking a time slot of their like. The company divides the territory into a number of smaller regions, and for each region a different time slot table is offered considering the expected demand of the region. Because the actual costs can only be evaluated once the demand is known, the problem becomes fairly complex as it involves the following three interconnected levels: the design of the time slot tables; customers choices of time slots; and the design of an optimized routing plan to serve customers requests. To approach this problem we propose a *large neighborhood search* algorithm, which uses as a repair method a mathematical formulation to bring solutions to a feasible state. To evaluate the expected cost of solutions we propose an algorithm that in a first step simulates customers choices based on one of a few simulation strategies and then, in a second step, evaluates the optimal routing plan for each operator. We create a set of instances based on historical data from the company, and test our approach under different demand scenarios. Computational results indicate that considerable improvements may be achieved with the implementation of optimized time slot tables.

5.1 Introduction

Attended home delivery is a last-mile service that requires the attendance of customers and it is an important distribution policy adopted in many industries, especially in e-grocery. The design of efficient and reliable delivery plans is a critical aspect for the success in this

sector. In addition, in order to meet customers needs in the best possible way, a certain level of service must be provided while balancing delivery costs. There are many examples of companies, such as Webvan and Streamline (see Agatz et al. 2011), that rose and fell because they were not able to design a sustainable distribution model.

Obviously, attended home delivery is not limited to e-grocers and appears in several other industries. As mentioned by Campbell and Savelsbergh (2005), this type of distribution is becoming common in drug companies that offer delivery services and companies that delivery office supplies to business customers. It is also common in the telecommunication sector, where companies such as AT&T allow customers to book appointments for cable and phone installation through their online channel.

In this chapter we approach a practical problem found in a large Italian corporate group called *Gruppo Iren*, that provides energy, gas and water services throughout the country. In particular, we study a problem faced by *Iren Emilia*, a subsidiary of this group that acts on the region of Emilia Romagna, Italy, providing gas and/or water services for approximately 1.300.000 habitants.

Following a series of laws from the European Union, starting in 1998 and culminating in the directive 2009/73/CE (see European Parliament 2009), the market of natural gas has been regulated and common rules have been established. As a consequence, all companies operating in this sector were forced by their national authorities to implement online applications in which customers are able to make requests for services by booking time slots. These time slots undergo national regulation constraints and are made available in certain days and time intervals.

Designing an automated approach that is capable of proposing cost-efficient time slot tables is a very complex task, because it involves solving a multi-level decision problem. The design of each time table has to take into account the expected demand so as the offer of slots is neither underestimated nor overestimated. Then, to evaluate the expected costs, customers have to first book their time slots, and only when the booking process is complete an optimized routing plan is created and the final costs are estimated.

To approach this problem we propose an optimization method based on the concept of *large neighborhood search* (LNS) and capable of producing time tables that incorporate all the aforementioned aspects. The quality of the solutions generated at each LNS iteration is evaluated by a procedure that in a first stage simulates customers choices on the system using up to four different simulation strategies, and then solves a routing subproblem involving multiple depots, multiple vehicles and strict time windows by means of a dedicated *integer linear programming* (ILP) model. In addition, our LNS uses as a repair method a second ILP model to efficiently bring solutions to a feasible state, while also optimizing the offer of slots in such a way that features that may have a negative impact on the routing plan are avoided.

The remainder of this chapter is organized as follows. A brief review of the related

literature is presented in Section 5.2 and the problem definition is provided in Section 5.3. The algorithm that we use to simulate customer choices and evaluate the expected solution costs is discussed in Section 5.4. Our overall solution approach is presented in Section 5.5 and computationally evaluated in Section 5.6. Finally, in Section 5.7 some conclusions are drawn and future research directions are presented.

5.2 Literature review

There are many challenges and opportunities in the context of attended home delivery. Agatz et al. (2008) use as an illustrative example one of the largest internet grocery stores in the U.S. at the time, named Peapod, to highlight combinatorial issues arising in attended home delivery and propose potential approaches to address them.

According to Campbell and Savelsbergh (2005), the fulfillment process of requests for most attended home delivery service models can be divided into three phases: 1. order capture and promise; 2. order sourcing and assembling; 3. order delivery. In our case, phase 1 may be divided into two steps: The first one concerns the definition of which and how many time slots to offer, and the second one refers to the choice of time slots by the customers. In phase 2 a routing plan is designed to minimize the expected routing costs, while in phase 3 requests are fulfilled.

There seem to be two main streams of research in the literature of attended home delivery. While there are those that focus on the aspect of management of service time windows, others emphasize on the routing and scheduling of delivery tours. Regarding the former, the literature focus either on the tactical or the operational levels (see, e.g., Ehmke and Campbell 2014). The tactical level is concerned with the design of the time windows itself, i.e., the number of time slots to offer, their length and decisions on whether or not they overlap. Punakivi and Saranen (2001) access the impact of variations on the length of time windows in both attended and unattended home deliveries. They show that significant cost reductions can be achieved in a scenario with completely flexible unattended services, compared to attended deliveries with two-hours time windows.

Campbell and Savelsbergh (2005) report that an expansion of time windows length from one hour to two hours can result in a 6% increase of the profits. This benefit is even greater when considering an expansion to three-hours time windows, reaching up to 11%. Of course there is an associated trade-off, that is, the longer the time windows the lesser the level of service that is provided.

On the other hand, the operational level refers to the selection of time slots by customers and decisions such as when to open and when to close certain time slots. In this sense, Campbell and Savelsbergh (2005) propose several mechanisms to determine when to accept or reject requests. In a later work, Campbell and Savelsbergh (2006) show that the use of

incentive polices to influence customers behavior can result in significant reductions in the delivery costs.

Perhaps the basic problem in the second stream of research is the *vehicle routing problem with time windows* (VRPTW), which has been extensively studied in the last two decades. For a deep review on the literature of this problem, we refer the interested reader to the surveys of Bräysy and Gendreau (2005a,b) and Baldacci et al. (2012).

Spliet and Gabor (2015) introduced the *time window assignment vehicle routing problem* (TWAVRP), where time windows have to be assigned to a fixed set of customers before the demand is known. Once the demand is revealed a routing plan is built with the objective of minimizing the travelling costs. They propose a branch-and-price algorithm with two types of route relaxation, and use capacity inequalities to strengthen their lower bound. Computational results indicate that their approach is able to solve instances with up to 25 customers and 3 demand realization scenarios. Our problem is different because the set of customers may vary consistently from one week to another, and because the time windows are not assigned by the company but chosen by the customers.

In the TWAVRP each customer requires a service in each scenario, while in the *consistent vehicle routing problem* (ConVRP), proposed by Groër et al. (2009), this constraint is relaxed. In addition, customers must always be visited by the same driver in each scenario and there is a limit on the total driving time. Recently, Spliet and Desaulniers (2015) proposed the *discrete time window assignment vehicle routing problem* (DTWAVRP). This problem differs from the TWAVRP by considering, for each customer, a finite number of candidate time windows from which a single one must be selected. The authors refer that this problem is common in the industry of retail chains and, indeed, have encountered it while collaborating with Dutch companies of the sector. They developed an exact branch-and-price algorithm, and five column generation heuristics that are used to solve large instances where the exact algorithm fails.

To our knowledge, the problem that most resembles ours is the one proposed by Agatz et al. (2011). They study a practical case found in the Netherlands based e-grocery company Albert.nl. For each zip-code the company must decide on a set of time slots to offer, while minimizing the delivery costs and ensuring an acceptable level of service to customers.

Prior to their study, the design of the time slot tables at Albert.nl was performed manually. Therefore, the authors aimed at developing a fully automated approach to create high quality time tables in a short amount of time. Customers can choose from two-hours time slots having a different fee depending on the day and time of the delivery. This is an interesting feature, common in the e-grocery industry, and allows for the manipulation of customer behavior. In fact, because of this, the authors suppose all time slots to be equality popular and demand to be evenly distributed across the offered slots. Another assumption is that the expected weekly demand for each zip-code is known, and it is independent of the time table offered. To solve the

problem, they propose a continuous approximation and an integer programming algorithm, where routing costs are computed by using a seed-based approach. From the analysis of the computational results, they present some useful insights, for instance: narrow delivery time windows, although convenient for the customers, can lead to an increase of up to 25% in routing costs, compared with the alternative of using two-hours time slots; differentiating the offered time table per regions can lead to up to 10% of cost savings; and optimization of time slot management decisions has a greater impact when the vehicle capacity allows it to serve several time slots consecutively. Our contribution is different from that of Agatz et al. (2011) because we consider more complex policies to simulate customer behaviors and more precise cost evaluations obtained by building actual vehicle routes. In addition, we cannot consider different fees for different time slots, but we need to take into account multi period distributions and penalties for late services.

5.3 Problem definition

In this section we formally introduce our optimization problem and present the notation used throughout the chapter. The problem is composed by three decision stages, which we describe in detail.

Stage 1: time table creation. The aim of this stage is to design the time tables that minimize unserviced demand and routing cost while satisfying capacity constraints. It is given a set R of regions, each having an expected required total service time. In our case study, services may require either half an hour or one hour, thus we model the expected time for a region $r \in R$ by an integer value q_r specifying the number of half hours of service required (i.e., if $q_r = 1$ only half an hour of service is required, if $q_r = 2$ an hour is required, and so on). From now on, let us call q_r the *expected demand* of region r. Let us also use the term basic time resource, or simply *resource* when no confusion arises, to define a time interval of half an hour.

The time horizon is divided into a set D of days, and each day is divided into a set T of non-overlapping time intervals. In our case study, $|R| = 12$ as shown in Figure 5.1, $|D| = 5$ (corresponding to 5 working days in a week, from Monday to Friday) and $|T| = 9$ (corresponding to intervals [8:30; 9:30], [9:30; 10:30], ..., [16:30, 17:30], where [12:30, 13:30] is reserved to lunch break). In the following a *time slot* is defined by the pair (d, t).

It is also given a set M of depots. Each region is associated univocally with a depot, so we can set $R = \cup_{i \in M} R_i$, where R_i is the subset of regions associated to depot $i \in M$. Each depot i has Q_i operators that are available to perform services and must start and end their routes at i. Each operator may satisfy σ resources per time slot. In our case study $|M| = 3$, and $\sigma = 2$ because all time slots last an hour.

A *time table* for a region r is an assignment of resources to the time slots. Let u_{rdt} be an

integer variable giving the number of resources assigned to region r and time slot (d, t). A stage 1 solution is a collection of time tables, one per region, satisfying the following condition

$$\sum_{r \in R_i} u_{rdt} \leq \sigma Q_i \qquad i \in M, d \in D, t \in T. \tag{5.1}$$

Stage 1 has two objectives: the primary one is to minimize the quantity of unserviced demand, whereas the secondary one is to minimize the routing cost. The values of these two objectives, for a given stage 1 solution, are computed in stage 2 and stage 3, respectively.

Stage 2: booking of time slots. At stage 2 the actual demand in a given time horizon is revealed. Let $\bar{u}_{rdt}$ define the stage 1 solution at hand, and let I denote the set of customers now requiring a service. This set is partitioned as $I = \cup_{r \in R} I_r$, where I_r is the subset of customers associated to region r. Each customer i requires a service of w_i resources (with w_i being equal to 1 or 2 in our case study), and attempts to book her/his *appointment* in the time table by choosing a time slot of her/his like according to the availability provided by the $\bar{u}_{rdt}$ values.

Let a_{irdt} be a binary variable taking the value 1 if customer i books her/his appointment in time slot (d, t) of region r. A stage 2 solution is a set of appointments satisfying the

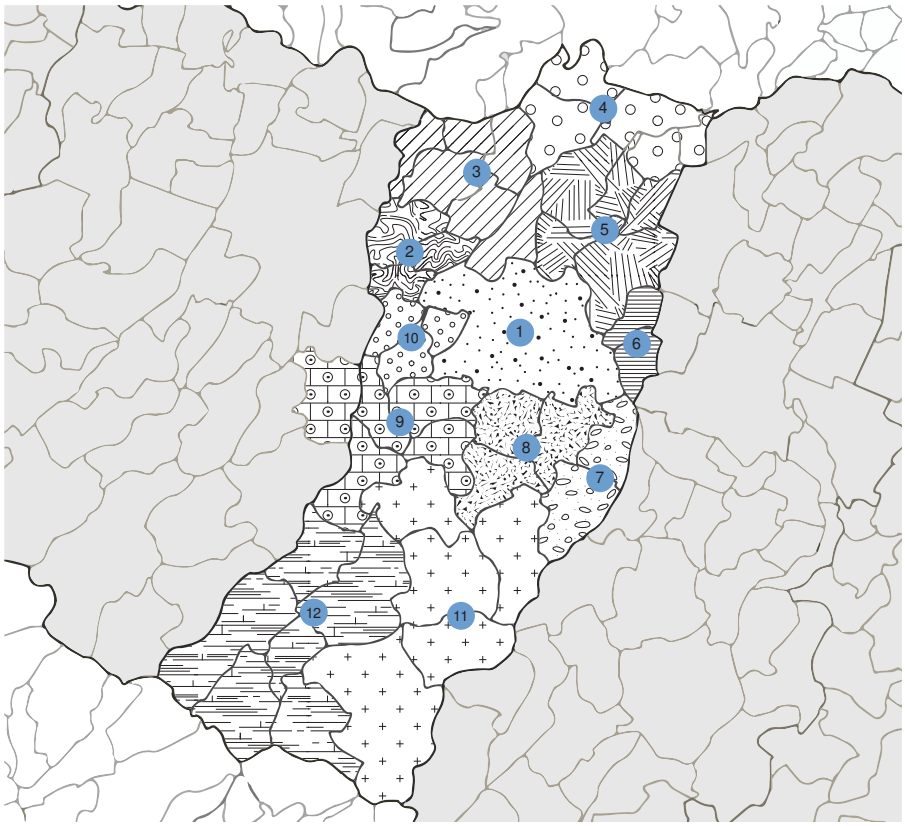


Figure 5.1: Division of the the territory of action into 12 non-overlapping regions.

following condition

$$\sum_{i \in I_r} w_i a_{irdt} \leq \bar{u}_{rdt} \qquad r \in R, d \in D, t \in T. \tag{5.2}$$

The decision is made by each customer, on a first-come first-served basis, independently from the will of other customers and of the company. To model the customers behavior in the most realistic way, we have thus developed a set of simulation strategies (discussed below in Section 5.4.1) that we use to set the values taken by the a_{irdt} variables.

Let us denote the total revealed demand of a region r by $W_r = \sum_{i \in I_r} w_i$. Notice that there is no guarantee that the total revealed demand is equal to the expected demand used in stage 1. In other words, W_r could be either greater than, equal to, or lower than q_r. Consequently, there can be cases in which demand is not fully satisfied. The cost of a stage 2 solution is indeed given by the sum of unserviced demands. This value is multiplied by a penalty parameter ρ, which is expressed in kilometers in our case study to make it directly comparable with the routing cost evaluated in stage 3.

Stage 3: design of the routing plan. Let $\bar{a}_{irdt}$ define the set of appointments computed in stage 2. Now let $G = (V, A)$ be a digraph, in which $V = I \cup M$ is the set of vertices and A is the set of arcs. Note that the graph is far from being complete, as A contains only arcs that connect vertices by satisfying the chronological order imposed by the appointments (in a way that is made clear in Section 5.4.2 below). Let c_{ij} be a non-negative routing cost associated to arc (i, j). The aim of stage 3 is to define a set of routes for the operators satisfying the following constraints:

- each route starts from a depot and returns to the same depot at the end of the services;
- at most Q_i routes are selected for each depot $i \in M$;
- each operator fulfills at most σ resources for each time slot;
- each appointment is serviced by an operator.

The cost of a stage 3 solution is given by the sum of the routing costs of the selected routes.

Notice that, in stage 3, an operator may perform a service in a region that is not associated with its depot if, by doing so, routing costs are reduced. This it is not considered in stage 1 to avoid creating time tables that reflect an unrealistic availability of operators, located too far from their depot to perform services at low cost.

An interesting detail is that, due to strict regulations imposed by the national authority on the sector, the company cannot influence on customers decisions by putting differentiated fees, applying penalties or giving incentives. Consequently, there may be time slots significantly more popular than others and many opportunities to reduce operational costs, common in other attended home delivery problems (see, e.g., Campbell and Savelsbergh 2006), are lost.

Furthermore, we note that the company is able to outsource a certain number of services through a third-party operator at a fixed cost per service, thus avoiding paying some of the penalties. With the aim of not complicating any further the problem, it is not in the scope of this chapter to consider decisions on which and how many services to outsource. Instead, we approximate outsourcing costs in a simple way by using the aforementioned penalty ρ for any unserved demand.

5.4 Evaluating the solution cost

To evaluate the expected cost of a given solution, the customers have to first book their time slots. Only then, routing and penalty costs can be estimated. This is a difficult task, because customers behaviors are not known beforehand and cannot be predicted with total accuracy. However, it is possible to simulate different scenarios according to a range of factors, including the use of historical data of the system usage. In this section we first propose four simulation strategies and then present an ILP model to evaluate the routing costs.

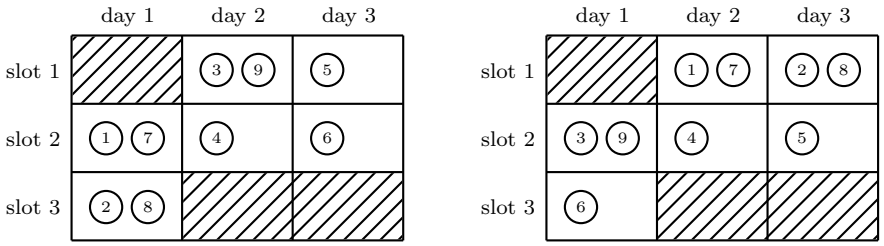
5.4.1 Simulation strategies

We are given a time table, decided in stage 1 for our approach, and a set of customers that require a service. The time horizon is a week (5 working days and 8 available time slots per day). The aim of the simulation process is to fix appointments in the time table by mimicking the customers behavior under reasonable assumptions.

The first two strategies that we adopted are called *evenly vertical* (EV) and *evenly horizontal* (EH). They are based on the assumption that all time slots are equally popular and services can be distributed evenly throughout the available time slots, as done in Agatz et al. (2011). At each iteration we choose a service and assign it to the next slot that has enough capacity to accommodate it. Except for the first iteration, that starts from the beginning of the time table, the others begin the search for a slot right after the one where the previous assignment has been made. Notice that, during the assignment of a service, it might happen that the end of the table is reached while searching for an available slot. In this case we return to the beginning of the time table and continue the search. In case there is no feasible slot to insert a service, we consider it as being unserved.

The difference between EV and EH lies on the criteria used for choosing the next slot. While EV performs a vertical search, looking at the remaining time slots on the same day before proceeding to the next day, in EH an horizontal search is used instead. As an illustrative example, consider a scenario where there are 9 services to be assigned into a small time table composed by just 3 days and 3 time slots, as the one in Figure 5.2-(a). For simplicity, consider that only non-hatched time slots are opened, each service requires 1 unit of resource and $\sigma = 2$. Figure 5.2-(a) shows the result obtained by using EV, and Figure 5.2-(b) the result obtained

by using EH.



(a) evenly vertical (b) evenly horizontal

Figure 5.2: Examples of simulation strategies *evenly vertical* (EV) and *evenly horizontal* (EH)

Our instances are based on historical data from the company, and information on the actual time slots for which services were performed is available. However, this information cannot be used by EV and EH. Our third strategy, called *rescheduling based* (RB), overcomes this issue by using information on the real assignments of services when simulating customers choices as follows. Let $(\tilde{d}, \tilde{t})$ be the time slot that was actually chosen in real life by a customer $i \in I$. At each iteration we select a customer i and try to assign her/his service to a time slot close to $(\tilde{d}, \tilde{t})$. In even iterations, we search for a feasible time slot $(\tilde{d}, t)$, i.e., we keep the day fixed but accept a different time. In odd iterations, we search for a time slot $(d, \tilde{t})$, i.e., at the same time but possibly in a different day. In case the process fails, we use the EV strategy to find a feasible time slot. If no time slot with enough capacity is available, then we mark the customer as unserved.

For the last strategy, called *popularity based* (PB), we first evaluated the popularity of each time slot on the period from 2010 to 2013. To this aim, we compute the frequency in which a time slot (d, t) was chosen, and then normalized the values, thus obtaining an array of probabilities p_{dt} such that $\sum_{d \in D} \sum_{t \in T} p_{dt} = 1$. At each iteration a customer $i \in I$ is chosen and then assigned to a time slot (d, t) having enough residual capacity, if any, according to probabilities p_{dt}.

For each unserved service a penalty ρ is paid. The resulting penalty cost is added to the routing cost, which is calculated when designing the routing plan.

5.4.2 Design of a routing plan

Having simulated customers choices, we must build a routing plan for each operator aiming at minimizing total travelling costs. The associated routing problem is decomposable by day, since the routing plan of one day is independent from those of the other days, and can be modeled as a variant of the *multidepot multiple traveling salesman problem* (MmTSP).

The MmTSP is a variation of the well known traveling salesman problem that consists in finding tours for salesmen starting from different depots in a such a way that all customers are visited exactly once and the total traveled distance is minimized. If salesmen are constraint to

end their routes at the same depot they originated from, the problem is called *fixed destination* MmTSP (f-MmTSP), otherwise it is referred as *non-fixed destination* MmTSP (nf-MmTSP). In the MmTSP literature, a maximum limit on the total distance traveled by each operator is also usually imposed, but this is not required in our problem. We refer the interested reader to Kara and Bektas (2006) and Bektas (2006).

At stage 3, customers had already chosen their appointments in the available time slots, so they must be visited in a way that respects the time windows constraints. Because these time windows are discretized in intervals that do not overlap, it suffices to solve the MmTSP under a *time-extended network* containing only those arcs that do not violate the time windows. To this aim, let us add to the original set of time slots two dummy slots, 0 and $|T|+1$ representing, respectively, the beginning and the end of a working day. The resulting set of time slots is thus $T' = \{0, 1, ..., |T|, |T| + 1\}$. Let I'_t be the set of customers that chose an appointment at time $t \in T$. Let us also duplicate each depot $i \in M$ in a set $M_i = \{i_0, i_1, ..., i_{|T|+1}\}$ of copy vertices, where each vertex i_t represents a visit to i at time t.

Now, let us define our time-extended network as a digraph $G' = (V', A')$, where $M' = \cup_{i \in M} M_i$ and $V' = I \cup M'$. The set A' of arcs is built by creating two types of arcs. The first type contains arcs (i, j) connecting each vertex $i \in I'_t$ to each vertex $j \in I'_{t+1}$, for $t \in T' \setminus \{|T| + 1\}$, ensuring that i and j are not two different depots. The second type contains arcs (i, j) connecting two customers $i, j \in I'_t$ (for $t \in T$) for which $w_i + w_j \leq \sigma$ holds.

An illustrative example is shown in Figure 5.3, where $M = \{1, 2\}$, $I = \{3, 4, 5, 6, 7, 8\}$, $T = \{1, 2, 3\}$ and $\sigma = 2$. All customers require two units of resource, except for customers 7 and 8 that require one unit each. Therefore, customers 7 and 8 are the only ones that can be served by the same operator in the same time slot.

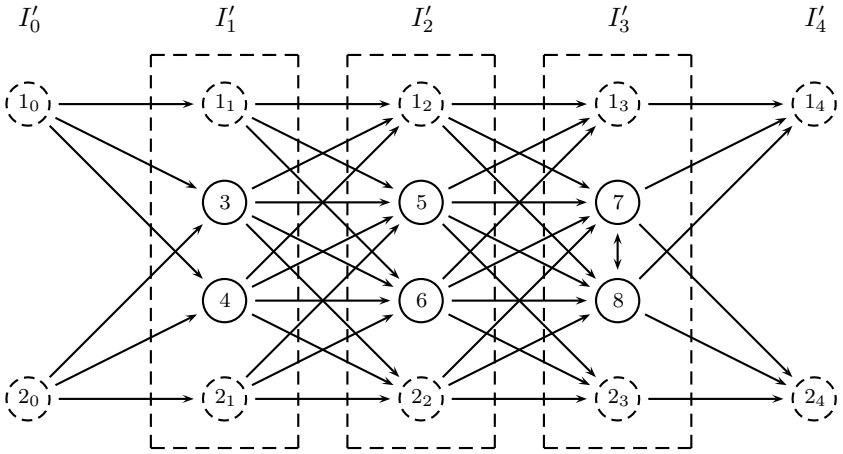


Figure 5.3: Example of a time-extended network with 2 depots (1 and 2), 6 customers and 3 time slots

Recall that c_{ij} is the routing cost of arc (i, j) and that each depot $i \in M$ has Q_i operators.

Let a binary variable x_{ij} take the value 1 if arc $(i,j) \in A'$ is used and 0 otherwise. The nf-MmTSP can be modeled as

$$\text{(nf-MmTSP)} \quad \min\ z_{nf} = \sum_{(i,j)\in A'} c_{ij}x_{ij} \tag{5.3}$$

subject to

$$\sum_{j\in V'} x_{i_0 j} \leq Q_i \qquad \forall\ i \in M \tag{5.4}$$

$$\sum_{i\in V'} x_{ij} = 1 \qquad \forall\ j \in V' \tag{5.5}$$

$$\sum_{j\in V'} x_{ij} - \sum_{j\in V'} x_{ji} = 0 \qquad \forall\ i \in V' \setminus \{i_0 : i \in M\} \setminus \{i_{|T|+1} : i \in M\} \tag{5.6}$$

$$\sum_{i\in \bar{S}} \sum_{j\in S} x_{ij} \geq \left\lceil \frac{\sum_{i\in S} w_i}{\sigma} \right\rceil \qquad \forall\ S \subseteq T'_t, \bar{S} \in T'_{t-1} : t \in T \tag{5.7}$$

$$x_{ij} \in \{0,1\} \qquad \forall\ (i,j) \in A' \tag{5.8}$$

The objective function (5.3) minimizes the total routing cost. Constraints (5.4) impose the maximum number of operators for each depot, whereas constraints (5.5) specify that all customers must be visited exactly once. Flow conservation is imposed by constraints (5.6), for all nodes except for the copies of the depots at times 0 and $|T|+1$. Subtour elimination is needed only for subsets of customers in the same time slot, and is obtained by means of constraints (5.7).

To adapt formulation (5.3)-(5.8) to the fixed problem variant, f-MmTSP, it suffices to add the following set of infeasible path constraints

$$\sum_{(i,j)\in\xi} x_{ij} \leq |\xi| - 1 \qquad \forall\ \xi \in \mathcal{R}, \tag{5.9}$$

where $\mathcal{R}$ is the set of all routes that begin and end at different depots.

Considering that this routing problem has to be solved every time we need to evaluate the quality of a solution for the main problem, efficiency is of great importance. We performed computational experiments with both variants and noticed a significant loss in performance with the inclusion of constraints (5.9) into the formulation, while the gap between the two solution values was usually very small. Therefore, we decided to use the nf-MmTSP to guide the search of our LNS.

5.5 Solution approach

The practical nature of the problem suggests the use of efficient heuristic approaches, able to provide good quality solutions in a reasonable amount of time. In addition, the use of exact

approaches is not straightforward in this case, given that there is no deterministic method to evaluate the cost of a solution with total accuracy.

An interesting feature of our problem is that the search space is very large. One must decide how many resources to offer for each time slot, day and region, while satisfying operational constraints. We note that, although it is usually fairly easy to find feasible solutions, it is computationally expensive to evaluate their expected cost, because, as described in Section 5.4, an ILP model is needed.

In this sense, we opted to use the metaheuristic LNS, originally proposed by Shaw (1998) and used later on with success in many routing problems (see e.g., Pisinger and Ropke (2010) and Dell'Amico et al. (2016)). This heuristic belongs to the class known as *very large scale neighborhood search* and, as the name suggests, it is capable of searching a large portion of the solution space. The LNS framework is usually composed by a *destroy* and a *repair* method. While the former destroys part of the solution, the latter is responsible for rebuilding feasibility. Algorithm 3 shows the pseudocode of our LNS. The algorithm receives in input the expected demand q_r of each region $r \in R$ (*regionsDemand*), an initial solution (*initialSolution*), the simulation strategy to be used (*simulationStrategy*), and the number of iterations to be elapsed (*numIt*). In lines 2 and 3 the local variables are initialized and the cost of the initial solution is evaluated. At each iteration of the main loop (line 4), a candidate solution (*newSolution*) is generated by destructing part of *bestSolution*, and then invoking the repair method (line 5). In case the newly created solution has a better expected cost than the incumbent, then *bestSolution* and *bestCost* are updated accordingly (line 7). The algorithm stops after *numIt* iterations and the incumbent solution is returned.

Algorithm 3 Pseudocode of our LNS algorithm

```
1: function LNS(regionsDemand, initialSolution, simulationStrategy, numIt)
2:     bestSolution ← initialSolution
3:     bestCost ← estimateSolutionCost(bestSolution, simulationStrategy)
4:     for it ← 0 to numIt do
5:         newSolution ← repair(destroy(bestSolution), regionsDemand)
6:         cost ← estimateSolutionCost(newSolution, simulationStrategy)
7:         if cost < bestCost then updateBestSolution(newSolution, cost)
8:     end for
9:     return bestSolution
10: end function
```

In the following we describe the proposed destroy and repair methods and comment on how to generate a feasible initial solution.

5.5.1 Destroy method

As mentioned by Pisinger and Ropke (2010), the choice of the destroy method is an important aspect for the efficacy of an LNS algorithm. If the degree of destruction is rather weak and only a small portion of the solution is destroyed, then the search space explored is considerably limited and the benefits of using such a heuristic may be lost. Similarly, if the degree of destruction is very high, then the heuristic tends to degrade into a repeated re-optimization process.

In our destroy method, for each region $r \in R$ we randomly select β opened time slots (i.e., β slots that have at least one unit of resource assigned) and empty them. Notice that if there are β or less opened time slots, then the table is totally destroyed. In our experiments, where each time table has a total of 40 time slots (5 days and 8 usable time slots per day), we found good results by using $\beta = 10$.

5.5.2 Repair method

Once a solution is partially or totally destroyed it must be rebuilt. To this end, we developed a repair method based on an ILP model. The basic idea is to use a *fitness function* that tries to maximize the number of serviced demands and the popularity of the chosen slots, but at the same time penalizes features that may have a negative impact on the routing plan.

Recall that $I_r \subseteq I$ is the set of customers associated with region r, that $p_{dt} \in [0, 1]$ defines the probability that a customer chooses a time slot (d, t) (see Section 5.4.1), and that variables u_{rdt} give the number of resources allocated to time slot (d, t) in region r (see Section 5.3). Let also $\ell \in T$ be the time slot dedicated to the lunch break.

During experimental analysis, we noticed that an evenly distribution of resources among consecutive time slots is a key factor in minimizing the routing costs. This allows operators to perform longer routes and make fewer trips to the depots. We incorporate this observation into the model by defining a penalty for unevenly distribution of resources among consecutive time slots. Let v_{rdt} be the number of displaced resources for a given time slot (d, t) of region r. This value is evaluated considering either a forward or a backward approach. In the forward approach, the number of displaced resources in (d, t) is $v_{rdt} = |u_{rdt} - u_{rd,t+1}|$, in other words, v_{rdt} gives the difference in the number of allocated resources between (d, t) and $(d, t+1)$. Similarly, in the backward approach $v_{rdt} = |u_{rdt} - u_{rd,t-1}|$ is the difference between the amount of resources allocated in (d, t) and $(d, t-1)$. Let γ be the penalty associated with each unit of displaced resource.

To model cases where the required demand cannot be entirely satisfied by the available operators, let z_r be an integer variable specifying the amount of demand that could not be allocated to region $r \in R$ due to capacity constraints. In addition, let Ω be a penalty value for each unassigned demand unit. We are now ready to introduce the following *three-index*

formulation (TIF).

$$\text{(TIF)} \quad \max \ f_{TIF} = \sum_{r\in R}\sum_{d\in D}\sum_{t\in T} p_{dt}u_{rdt} - \sum_{r\in R}\sum_{d\in D}\sum_{t\in T}\gamma\ v_{rdt} - \sum_{r\in R}\Omega z_r \tag{5.10}$$

$$\text{subject to}$$

$$\sum_{d\in D}\sum_{t\in T} u_{rdt} = q_r - z_r \qquad \forall\ r \in R \tag{5.11}$$

$$\sum_{r\in R_i} u_{rdt} \leq \sigma Q_i \qquad \forall\ i \in M, d \in D, t \in T \tag{5.12}$$

$$v_{rdt} \geq u_{rdt} - u_{rd,t+1} \qquad \forall\ r \in R, d \in D, t \in T \setminus \{\ell - 1, \ell, |T|\} \tag{5.13}$$

$$v_{rdt} \geq u_{rd,t+1} - u_{rdt} \qquad \forall\ r \in R, d \in D, t \in T \setminus \{\ell - 1, \ell, |T|\} \tag{5.14}$$

$$v_{rdt} \geq u_{rdt} - u_{rd,t-1} \qquad \forall\ r \in R, d \in D, t \in \{\ell - 1, |T|\} \tag{5.15}$$

$$v_{rdt} \geq u_{rd,t-1} - u_{rdt} \qquad \forall\ r \in R, d \in D, t \in \{\ell - 1, |T|\} \tag{5.16}$$

$$u_{rd\ell} = 0 \qquad \forall\ r \in R, d \in D \tag{5.17}$$

$$u_{rdt} \geq 0, integer \qquad \forall\ r \in R, d \in D, t \in T \tag{5.18}$$

$$v_{rdt} \geq 0, integer \qquad \forall\ r \in R, d \in D, t \in T \tag{5.19}$$

$$z_r \geq 0, integer \qquad \forall\ r \in R \tag{5.20}$$

The fitness function (5.10) maximizes the serviced demand and the aggregated value of popularity per allocated resource and, at the same time, tries to avoid unevenly distribution of resources. Constraints (5.11) specify that the number of allocated resources per region is equal to the served demand. Recall that each region is associated with a single depot and each operator can perform at most σ resources per time slot. Constraints (5.12) ensure that the number of resources allocated to each time slot does not exceed the total operator capacity. Constraints (5.13) and (5.14) evaluate the penalties for unevenly distribution of resources using a forward approach. Notice that this approach is not feasible for the time slot just before the lunch break and the last one of the day. For these slots, a backward approach is used instead in (5.15) and (5.16). Constraints (5.17) impose that time slots associated with the lunch break are always closed. Finally, constraints (5.18)-(5.20) formally define variables u, v and z.

Through computational experiments, formulation TIF has proved to be a good tool to generate feasible solutions for the problem. Thus it is first used as a method for generating initial solutions for the LNS algorithm. Its main use is, however, as a repair method. Because the main objective of the repair method is that of bringing solutions to a feasible state, but not optimizing the whole scenario, we impose in TIF additional constraints specifying that the number of resources allocated to time slots not altered during the call to the destroy method can only increase. Let Θ_r be the set of time slots not destroyed by the destroy procedure in region $r \in R$ and b_{rdt} the amount of resources in time slot $(d, t) \in \Theta_r$. Then, at every call to

the repair method we add to TIF the following constraint

$$u_{rdt} \geq b_{rdt} \qquad \forall\, r \in R, d \in D, t \in T : (d,t) \in \Theta_r \tag{5.21}$$

5.5.3 Quality of service

In attended home delivery it is important to establish a certain level of *quality of service* (QoS) in the time tables offered to customers. As mentioned by Agatz et al. (2011), an important aspect in this sense is related to the balance on the distribution of resources between morning and afternoon time slots along the week. Morning slots are those with an index in T less than ℓ, and afternoon slots are those with an index greater than ℓ. Let α_m and α_a be, respectively, the minimum percentage of resources that should be allocated to morning and afternoon slots, satisfying $\alpha_m + \alpha_a \leq 1$. We first impose a high QoS by embedding into TIF the following constraints

$$\sum_{t=1}^{\ell-1} \sum_{d \in D} u_{rdt} \geq \alpha_m q_k \qquad \forall\, r \in R \tag{5.22}$$

$$\sum_{t=\ell+1}^{|T|} \sum_{d \in D} u_{rdt} \geq \alpha_a q_k \qquad \forall\, r \in R \tag{5.23}$$

The second additional QoS constraint that we consider imposes a limit g on the maximum number of consecutive days without any time slots opened in a region. This is particularly useful for time tables of regions with low demand. Let y_{rd} be a binary variable indicating whether or not there is at least one time slot opened in day $d \in D$ for region $r \in R$. In addition, let $\phi(d,g) = \{(i \bmod |D|) + 1 : i = d-1, d, \ldots, d+g-1\}$. For instance, if $D = \{1, ..., 5\}$, $d = 4$ and $g = 2$, then $\phi(4,2) = \{4, 5, 1\}$. Then, a high QoS is imposed by also adding to TIF the following constraints,

$$\sum_{t \in T} x_{rdt} \geq y_{rd} \qquad \forall\, r \in R, d \in D \tag{5.24}$$

$$\sum_{l \in \phi(d,g)} y_{rl} \geq 1 \qquad \forall\, r \in R, d \in D \tag{5.25}$$

$$y_{rd} \in \{0,1\} \qquad \forall\, r \in R, d \in D \tag{5.26}$$

From now on we refer to the formulation composed by (5.10)-(5.20) and (5.22)-(5.26) as TIF$^+$.

5.6 Computational experiments

Algorithms were implemented in C++ and the computational experiments were run on a PC with an Intel Core i7-3770 3.40 GHz with 8 Gb of RAM. Cplex 12.6.2 was used to solve the ILP models with 8 threads and the default options. We used a set of 52 benchmark instances referring to the weeks of 2012, year in which the company first introduced the online application to customers. These instances include real distance values, computed with

a *geographical information system* software using a map from OpenStreetMap.

Our aim is to provide automatic solutions and compare with the ones used by the company. However, it is difficult to accurately evaluate their solution because they currently use a fixed low base estimation of the demand, and then perform manual additions of resources to the time slots on the fly when needed. To better understand the evolution of the demand along the year, we performed an analysis on the services performed in 2012. Results are shown in Figure 5.4. The x-axis reports the 2012 weeks, and the y-axis the aggregated amount of resources (i.e., $\sum_{r \in R} q_r$). Let *base* be the base estimation of demand by the company in number of resources, consider *2012* as the weekly demand of resources during 2012, and let *avg 2010-2013* be the average monthly demand during the period from 2010 to 2013. As one can see, *2012* is usually quite close to *avg 2010-2013*, apart from a few weeks with high difference. The base value adopted by the company is indeed almost coincident with the demands at the beginning of January and middle August, but very different in other weeks, especially during periods with peak demand.

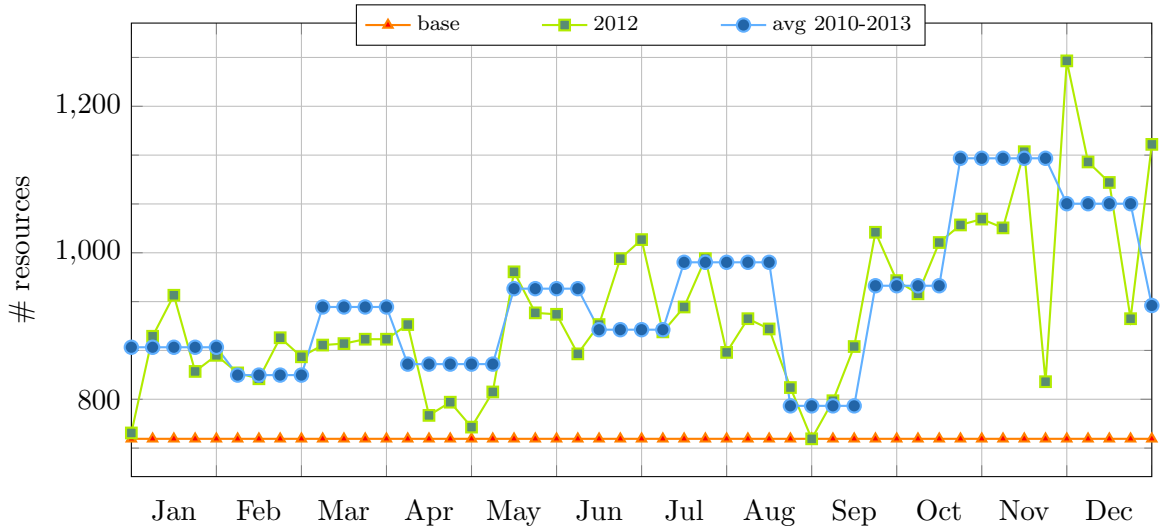


Figure 5.4: Evolution of the demand in 2012, compared with base demand used by the company and average demand in 2010-2013.

The expected demand of each region is a key information in the LNS, because it is used by formulation TIF to rebuild solutions, and directly influence the number of resources offered in the time tables. In our experiments we tested the LNS under three scenarios. In the first one, namely *base demand*, we set q_r, for $r \in R$, to the same number of resources as offered in the time tables currently employed at the company. The second scenario, referred as *avg demand*, considers instead q_r to be the average demand per month, evaluated considering the period from 2010 to 2013. Lastly, we consider an *utopic* scenario, namely *a priori demand*, in which the number of resources requested for each week is known beforehand and time tables are built accordingly. As a consequence, it is expected that in the third scenario unserved

demand is minimized.

Recall that we assume a penalty ρ to be paid for each service that is not assigned during the simulation of customers choices. We set $\rho = 20$ on the basis of considerations from the real world application. Later we will present a sensitivity analysis regarding this parameter. In addition, we set the penalty value $\Omega = 100$ to minimize unassigned demand in formulation TIF.

In Tables 5.1, 5.2, 5.3 and 5.4 we present the results of a monthly analysis during 2012 for the strategies EV, EH, RB and PB, respectively. We evaluate our algorithm LNS under the three demand scenarios and compare the results with the evaluation of the solution currently used by the company. The expected cost of the company's solution, shown in column *current*, was evaluated taking the company time tables and evaluating the expected costs by running the algorithm presented in Section 5.4. Notice that this does not take into account possible manual adjustments. Columns named *cost* specify the average expected cost in kilometers of the time tables, while columns named *gap* specify the improvement of the respective approach over the value in column *current*. In columns named *sec* we show average computational times in seconds.

Results show that our LNS is able to achieve significant improvements in a reasonable amount of time. These improvements range in the intervals 3.4%-14.2% for EV, 5.6%-15.6% for EH, 5.2%-15.7% for RB and 6.3%-14.2% for PB. They are quite consistent, showing that the algorithm is robust and it is able to adapt to different situations. Notice that during high demand periods, such as from September to December, improvements are the highest. This is explained by the elevated number of requests per time slot in certain regions, which allows for greater routing options and a better optimization of the routing plan. An important aspect in this regard is the fact that formulation TIF usually leads to an evenly distribution of resources in sequential time slots.

Table 5.1: Evaluation of LNS for the simulation strategy EV

		current	LNS - base demand			LNS - avg demand			LNS - a priori demand		
month	#	cost	cost	gap	sec	cost	gap	sec	cost	gap	sec
Jan	5	5878.9	5563.3	-5.4%	278.3	5469.0	-7.0%	127.9	5225.4	-11.1%	259.4
Feb	4	5446.4	5274.5	-3.2%	273.1	5002.4	-8.2%	280.0	4882.5	-10.4%	251.7
Mar	4	6110.0	6004.2	-1.7%	282.4	5463.0	-10.6%	274.0	5416.0	-11.4%	284.4
Apr	5	5090.8	4863.3	-4.5%	275.5	4863.3	-4.5%	281.4	4703.0	-7.6%	262.9
May	4	6648.9	6326.6	-4.8%	278.8	5706.2	-14.2%	291.3	5547.7	-16.6%	299.3
Jun	4	6908.0	6724.4	-2.7%	281.6	6285.5	-9.0%	284.7	5803.2	-16.0%	309.2
Jul	5	6570.5	6365.0	-3.1%	286.9	5788.1	-11.9%	290.3	5671.9	-13.7%	290.7
Aug	4	4887.8	4545.9	-7.0%	271.8	4623.2	-5.4%	269.1	4393.6	-10.1%	285.6
Sep	4	7212.1	6989.2	-3.1%	281.5	6355.5	-11.9%	207.4	5923.6	-17.9%	259.5
Oct	5	7672.0	7458.9	-2.8%	281.6	6487.0	-15.4%	49.6	6398.4	-16.6%	217.3
Nov	4	8835.2	8710.6	-1.4%	280.9	7285.4	-17.5%	220.8	7036.3	-20.4%	163.3
Dec	4	6659.1	6617.4	-0.6%	269.7	5919.5	-11.1%	280.4	5427.2	-18.5%	192.1
avg		6493.3	6287.0	-3.4%	278.5	5770.7	-10.6%	238.1	5535.7	-14.2%	256.3

Table 5.2: Evaluation of LNS for the simulation strategy EH

		current	LNS - base demand			LNS - avg demand			LNS - a priori demand		
month	#	cost	cost	gap	sec	cost	gap	sec	cost	gap	sec
Jan	5	6002.0	5592.3	-6.8%	283.1	5472.7	-8.8%	174.2	5253.4	-12.5%	269.3
Feb	4	5687.7	5323.2	-6.4%	278.2	5131.1	-9.8%	280.3	4951.1	-12.9%	268.0
Mar	4	6339.6	6056.2	-4.5%	280.9	5570.8	-12.1%	277.1	5472.7	-13.7%	286.0
Apr	5	5297.6	4968.1	-6.2%	280.5	4970.1	-6.2%	281.8	4804.4	-9.3%	282.7
May	4	6828.2	6435.4	-5.8%	284.8	5880.1	-13.9%	289.5	5634.0	-17.5%	277.2
Jun	4	7257.1	6801.1	-6.3%	283.4	6433.5	-11.3%	285.2	5899.4	-18.7%	294.6
Jul	5	6825.0	6441.4	-5.6%	286.0	5991.0	-12.2%	289.9	5805.1	-14.9%	272.9
Aug	4	5120.6	4625.3	-9.7%	277.1	4803.0	-6.2%	282.0	4519.9	-11.7%	275.6
Sep	4	7402.3	7075.5	-4.4%	282.4	6479.2	-12.5%	241.6	6043.2	-18.4%	219.3
Oct	5	7901.4	7516.5	-4.9%	282.5	6601.9	-16.4%	51.7	6529.1	-17.4%	186.6
Nov	4	8976.3	8719.4	-2.9%	282.4	7450.0	-17.0%	198.2	7078.2	-21.1%	139.8
Dec	4	6939.1	6693.9	-3.5%	276.0	6065.8	-12.6%	280.6	5584.6	-19.5%	167.6
avg		6714.7	6354.0	-5.6%	281.4	5904.1	-11.6%	244.3	5631.2	-15.6%	245.0

Table 5.3: Evaluation of LNS for the simulation strategy RB

		current	LNS - base demand			LNS - avg demand			LNS - a priori demand		
month	#	cost	cost	gap	sec	cost	gap	sec	cost	gap	sec
Jan	5	6167.3	5773.2	-6.4%	275.7	5777.6	-6.3%	115.8	5375.3	-12.8%	269.1
Feb	4	5834.0	5576.8	-4.4%	272.6	5312.2	-8.9%	282.7	5109.6	-12.4%	252.0
Mar	4	6635.5	6298.2	-5.1%	283.4	5891.1	-11.2%	252.9	5619.9	-15.3%	280.7
Apr	5	5650.9	5203.3	-7.9%	264.9	5370.4	-5.0%	278.8	5071.9	-10.2%	270.4
May	4	7168.7	6665.0	-7.0%	273.4	6106.3	-14.8%	293.5	5762.1	-19.6%	268.2
Jun	4	7413.4	7087.7	-4.4%	276.8	6764.1	-8.8%	276.0	6245.9	-15.7%	295.3
Jul	5	6976.1	6684.8	-4.2%	281.2	6355.2	-8.9%	278.2	6017.6	-13.7%	242.2
Aug	4	5362.3	4995.6	-6.8%	270.1	5113.6	-4.6%	275.9	4846.9	-9.6%	274.4
Sep	4	7784.7	7363.1	-5.4%	263.6	6791.7	-12.8%	238.2	6165.0	-20.8%	234.0
Oct	5	8071.2	7788.8	-3.5%	274.8	7001.0	-13.3%	54.2	6653.3	-17.6%	196.0
Nov	4	9246.2	8909.8	-3.6%	279.8	7797.0	-15.7%	184.2	7293.6	-21.1%	152.0
Dec	4	7212.3	6935.9	-3.8%	258.5	6369.9	-11.7%	280.8	5826.0	-19.2%	174.2
avg		6960.2	6606.8	-5.2%	272.9	6220.8	-10.2%	234.3	5832.3	-15.7%	242.4

Table 5.5 presents the average results of all four simulation strategies to provide a clear overview. Overall, the results indicate that improvements of about 5% in average can be achieved in the most restrictive demand scenario, while in the *a priori* case, where the demand is known beforehand, these improvements reach up to 15%, in average.

Furthermore, we performed additional experiments and noticed that, giving the LNS the time tables used by the company as initial solution resulted in a small but consistent worsening of the solution costs from 2.7% up to 5%. This shows that using TIF as the method for generating initial solutions is more efficient. We also ran a version of the LNS, in which the solution used at each iteration is not the incumbent, but the one provided by the previous iteration. This led to a worsening in solution costs of about 1%.

Clearly, *a priori demand* cannot be applied in practice, but *avg demand* seems a good

Table 5.4: Evaluation of LNS for the simulation strategy PB

		current	LNS - base demand			LNS - avg demand			LNS - a priori demand		
month	#	cost	cost	gap	sec	cost	gap	sec	cost	gap	sec
Jan	5	6100.3	5683.1	-6.8%	269.1	5695.0	-6.6%	151.2	5464.0	-10.4%	273.4
Feb	4	5907.2	5551.1	-6.0%	276.0	5341.3	-9.6%	281.5	5187.7	-12.2%	255.9
Mar	4	6544.6	6212.7	-5.1%	258.4	5924.4	-9.5%	268.1	5644.1	-13.8%	282.3
Apr	5	5665.5	5152.1	-9.1%	268.8	5334.3	-5.8%	283.1	5130.6	-9.4%	258.1
May	4	7023.4	6577.5	-6.3%	278.9	6073.9	-13.5%	292.6	5883.1	-16.2%	270.2
Jun	4	7409.2	6882.4	-7.1%	276.0	6719.9	-9.3%	263.6	6310.9	-14.8%	246.3
Jul	5	7016.8	6541.2	-6.8%	280.5	6400.1	-8.8%	267.2	6157.2	-12.3%	264.1
Aug	4	5336.3	4812.7	-9.8%	272.4	4987.3	-6.5%	281.8	4866.9	-8.8%	279.2
Sep	4	7607.8	7227.2	-5.0%	274.4	6683.1	-12.2%	239.0	6312.7	-17.0%	225.5
Oct	5	8082.3	7672.5	-5.1%	280.5	7013.5	-13.2%	52.5	6708.9	-17.0%	193.5
Nov	4	9074.1	8759.6	-3.5%	279.2	7753.6	-14.6%	244.6	7295.7	-19.6%	144.5
Dec	4	7125.9	6763.4	-5.1%	272.7	6256.0	-12.2%	284.6	5750.6	-19.3%	168.9
avg		6907.8	6486.3	-6.3%	273.9	6181.9	-10.2%	242.5	5892.7	-14.2%	238.5

Table 5.5: Average results of LNS for the the four simulation strategies

		current	LNS - base demand			LNS - avg demand			LNS - a priori demand		
month	#	cost	cost	gap	sec	cost	gap	sec	cost	gap	sec
Jan	5	6037.1	5653.0	-6.4%	276.5	5603.6	-7.2%	142.3	5329.5	-11.7%	267.8
Feb	4	5718.8	5431.4	-5.0%	275.0	5196.7	-9.1%	281.1	5032.7	-12.0%	256.9
Mar	4	6407.4	6142.9	-4.1%	276.3	5712.3	-10.8%	268.0	5538.2	-13.6%	283.4
Apr	5	5426.2	5046.7	-7.0%	272.4	5134.5	-5.4%	281.3	4927.5	-9.2%	268.5
May	4	6917.3	6501.1	-6.0%	279.0	5941.6	-14.1%	291.7	5706.7	-17.5%	278.7
Jun	4	7246.9	6873.9	-5.1%	279.5	6550.8	-9.6%	277.4	6064.9	-16.3%	286.4
Jul	5	6847.1	6508.1	-5.0%	283.7	6133.6	-10.4%	281.4	5912.9	-13.6%	267.5
Aug	4	5176.7	4744.9	-8.3%	272.9	4881.8	-5.7%	277.2	4656.8	-10.0%	278.7
Sep	4	7501.7	7163.7	-4.5%	275.5	6577.4	-12.3%	231.6	6111.1	-18.5%	234.5
Oct	5	7931.7	7609.2	-4.1%	279.9	6775.8	-14.6%	52.0	6572.4	-17.1%	198.3
Nov	4	9033.0	8774.9	-2.9%	280.6	7571.5	-16.2%	212.0	7176.0	-20.6%	149.9
Dec	4	6984.1	6752.7	-3.3%	269.2	6152.8	-11.9%	281.6	5647.1	-19.1%	175.7
avg		6769.0	6433.5	-5.1%	276.7	6019.4	-10.6%	239.8	5723.0	-14.9%	245.5

compromise. It models fluctuations on the demand and allows for cost savings of about 10%. Figure 5.5 provides a graphical visualization of the results in Table 5.5 and evidentiate these conclusions.

In Table 5.6, we present the results of an analysis on the impact of imposing a certain quality of service (QoS) for the design of time tables. In this case, formulation TIF$^+$ is used to both provide an initial solution and repair destroyed solutions, and we set $\alpha_m = 0.3$, $\alpha_a = 0.3$ and $gap = 2$. Thus, time tables produced by this approach, namely LNS_{QoS}, impose a minimum of 30% of resources allocated to morning time slots, another 30% to afternoon slots and also guarantee that there cannot be more than 2 consecutive days without any offer of time slots in any region. The first column of each approach refers to the expected average

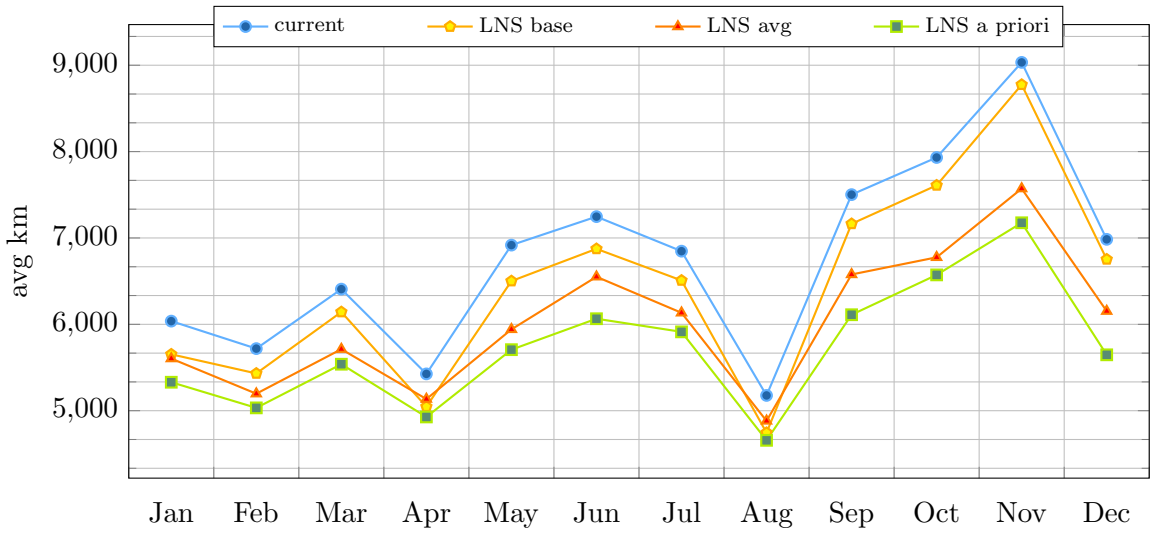


Figure 5.5: Average results of LNS for the the four simulation strategies in a chart

monthly cost of solutions, while the second column of LNS_{QoS} specify the average increase in the solution cost with respect to the LNS on the same scenario.

Table 5.6: Impact of imposing QoS constraints on the expected solution cost

		base demand			avg demand			a priori demand		
		LNS	LNS_{QoS}		LNS	LNS_{QoS}		LNS	LNS_{QoS}	
month	#	cost	cost	gap	cost	cost	gap	cost	cost	gap
Jan	5	5653.0	5880.8	4.0%	5603.6	5774.7	3.1%	5329.5	5608.5	5.2%
Feb	4	5431.4	5636.5	3.8%	5196.7	5273.2	1.5%	5032.7	5199.8	3.3%
Mar	4	6142.9	6328.0	3.0%	5712.3	5944.2	4.1%	5538.2	5786.7	4.5%
Apr	5	5046.7	5241.6	3.9%	5134.5	5254.6	2.3%	4927.5	5130.0	4.1%
May	4	6501.1	6734.5	3.6%	5941.6	6208.6	4.5%	5706.7	5990.8	5.0%
Jun	4	6873.9	7024.9	2.2%	6550.8	6784.2	3.6%	6064.9	6382.3	5.2%
Jul	5	6508.1	6733.2	3.5%	6133.6	6314.3	2.9%	5912.9	6244.6	5.6%
Aug	4	4744.9	4980.0	5.0%	4881.8	5009.5	2.6%	4656.8	4852.1	4.2%
Sep	4	7163.7	7380.5	3.0%	6577.4	6743.6	2.5%	6111.1	6374.7	4.3%
Oct	5	7609.2	7817.1	2.7%	6775.8	6995.6	3.2%	6572.4	6815.6	3.7%
Nov	4	8774.9	8947.4	2.0%	7571.5	7691.1	1.6%	7176.0	7489.6	4.4%
Dec	4	6752.7	6866.2	1.7%	6152.8	6386.6	3.8%	5647.1	5793.2	2.6%
avg		6433.5	6630.9	3.2%	6019.4	6198.3	3.0%	5723.0	5972.3	4.3%

As expected, these results indicate that imposing QoS constraints has a negative impact on the solution cost, but it is quite limited. The average increase in cost is about 3.5% and it is surprisingly stable over the three scenarios. Despite the additional requirements, our LNS still achieves significant improvements over the time tables used by the company. In the first scenario LNS_{QoS} shows 2.1% of average improvements, while in the second and third, these values reach up to 8% and 11.2%, respectively.

We now present a sensitivity analysis on the parameter ρ, responsible for establishing a

penalty value for unserved demand. For this analysis we considered $\rho \in \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$. Results are presented in Figure 5.6. Notice that, except for scenario *base demand*, the higher is the penalty value the more our LNS can improve over the time tables used by the company. This is to be expected because, as seen in Figure 5.4, the company is currently using low expected demand values. Indeed, the average number of unserved resources in their current time tables ranges around 100, in average, while in our LNS this value is reduced to about 36 and 14, in average, for scenarios *avg demand* and *a priori demand*, respectively. Furthermore, the results seen for *base demand* is not a surprise, because this scenario considers the same expected demand as the company's time tables.

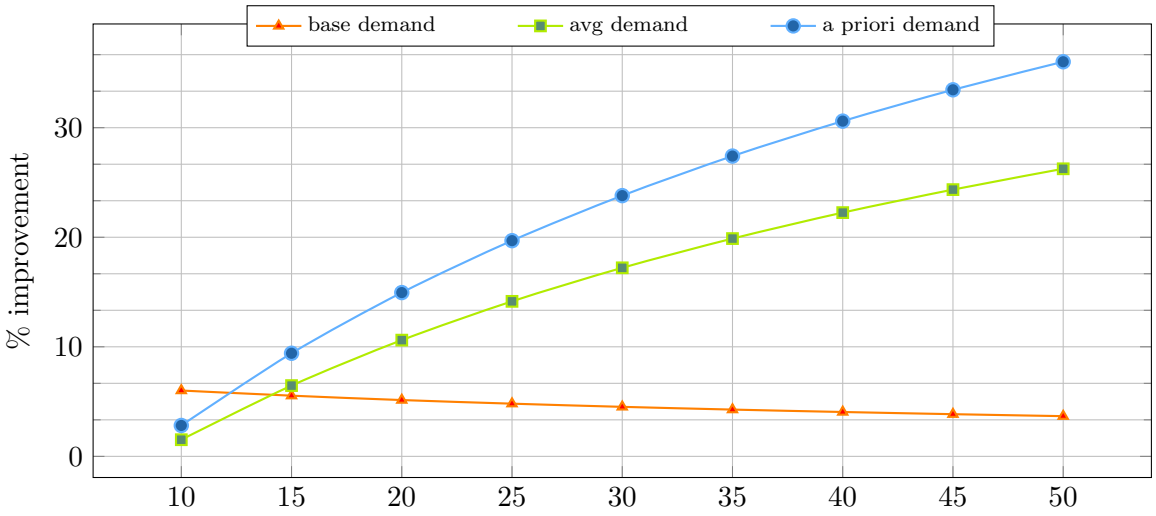


Figure 5.6: Sensitivity analysis on the value of ρ.

5.7 Conclusions

We studied a practical attended home delivery problem found in the context of an Italian service provider. Starting from 2012, this company provides an online application in which customers can request services by choosing a time slot of their like.

The territory of action is divided into 12 regions, each having a dedicated time slot table that varies according to the associated expected demand. Currently, these time tables are manually built based on historical information of service performance, and changes are implemented as needed. Developing an automated approach to produce cost efficient time tables involves a complex multi level decision problem. Time tables have to be built considering a certain expected demand, then customers choices are simulated and an assignment plan is designed. Lastly, routes for each available operator are designed to minimize routing costs. Only then the quality of the time tables can be assessed. In this sense, the main objective of this chapter was to propose an effective approach that allows for the creation of time tables

that are cost efficient and adapt to the varying demand along the year.

One of the most critical aspects of this problem refers to how to evaluate the expected cost of a solution, given that demand is not known beforehand and it is a key component to define the routing costs. We proposed an LNS algorithm that, uses as a repair method an ILP model to bring solutions to a feasible state, while optimizing the distribution of resources in such a way good features are preserved. Then we simulate customers choices of time slots using one of four simulation strategies. Having the assignment plan, a second ILP model is used to estimate the associated routing costs.

We performed computational experiments on three different demand scenarios. In the first, we considered the same expected demand as in the time tables implemented at the company. This is the most restrictive scenario because it completely undermines demand fluctuations that are common in practice. The second scenario considers instead the average monthly demand to propose more flexible and adaptable time tables, while the third considers an utopic case where demand is known, but not customers choices. Results indicate that average improvements of about 5%, 10% and 15% can be achieved at each scenario, respectively.

As a future research direction, it would be interesting to study a rolling horizon approach to explore the dynamic aspects of the problem and evaluate the trade-off of postponing the performance of some services instead of outsourcing. Another potential development is to evaluate the impact of changing the arrangement of regions, which can lead to further improvements.

5.8 Bibliography

N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh. Challenges and opportunities in attended home delivery. *Operations Research/ Computer Science Interfaces Series*, 43: 379–396, 2008.

N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh. Time Slot Management in Attended Home Delivery. *Transportation Science*, 45(3):435–449, aug 2011.

R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

T. Bektas. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, 2005a.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005b.

A. Campbell and M. Savelsbergh. Decision Support for Consumer Direct Grocery Initiatives. *Transportation Science*, 39(3):313–327, 2005.

A. Campbell and M. Savelsbergh. Incentive Schemes for Attended Home Delivery Services. *Transportation Science*, 40(3):327–341, aug 2006.

M. Dell'Amico, M. Iori, S. Novellani, and T. Stützle. A Destroy and Repair Algorithm for the Bike sharing Rebalancing Problem. *Computers & Operations Research*, 2016. . Forthcoming.

J. F. Ehmke and A. Campbell. Customer acceptance mechanisms for home deliveries in metropolitan areas. *European Journal of Operational Research*, 233(1):193–207, feb 2014.

European Parliament. Directive 2009/73/EC Of The European Parliament And Of The Council, 2009. Available at `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:211:0094:0136:en:PDF`.

C. Groër, B. Golden, and E. Wasil. The Consistent Vehicle Routing Problem. *Manufacturing & Service Operations Management*, 11(4):630–643, 2009.

I. Kara and T. Bektas. Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*, 174(3):1449–1458, 2006.

D. Pisinger and S. Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer, 2010.

M. Punakivi and J. Saranen. Identifying the success factors in e-grocery home delivery. *International Journal of Retail & Distribution Management*, 29(4):156–163, 2001.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems, 1998.

R. Spliet and G. Desaulniers. The discrete time window assignment vehicle routing problem. *European Journal of Operational Research*, 244(2):379–391, 2015.

R. Spliet and A. F. Gabor. The Time Window Assignment Vehicle Routing Problem. *Transportation Science*, 49(4):721–731, nov 2015.

Appendices

Appendix A

On the selection of the best supplier for a global service provider

In this chapter we study a practical group decision making problem faced by an Italian company acting on the sector of facility management. We aim at developing a decision support system to aid decision makers at the company, and present the methodology used to derive weights to the criteria used to evaluate which are the most suitable suppliers to assign to clients contracts.

A.1 Introduction

In general, *multi-criterion decision making* (MCDM) is an important subject for many companies. It is perhaps the most well know branch of decision making and deals with optimization problems involving alternatives that are evaluated under multiple objectives. There are several MCDM methods in the literature to support *decision makers* (DMs), but the *analytic hierarchy process* (AHP) is perhaps one of most widely used. The AHP is a method based on mathematics and psychology that enables DMs to organize and analyze complex decision making scenarios in order to make decisions. Since it was introduced by Saaty (1977), it has been successfully applied to a variety of problems and reported in several hundreds of papers in the literature, for instance: supplier selection (Wang and Yang 2009, Chamodrakas et al. 2010, Labib 2011), staff recruitment (Celik et al. 2009, Khosla et al. 2009), drugs selection (Vidal et al. 2010) and university evaluation (Lee 2010). For a comprehensive survey we refer the interested reader to Ishizaka and Labib 2011.

In the AHP, the decision problem is first decomposed into a hierarchy of subproblems that can be analyzed independently. One of the strongest features of this method is that the hierarchy may be composed by any aspect of the decision problem, from elements that can be carefully measured to those which are very subjective and highly dependent on personal views.

Psychologists argue that it is easier and more accurate to express opinions on only two alternatives at a time than simultaneously on all alternatives (see, e.g., Ishizaka and Labib 2009). Indeed, in the AHP, instead of rating elements by their order of importance all together, a pair-wise approach is used, comparing two elements at a time using a ratio scale that requires no unit of measurement. Based on this information, with the AHP we are able to derive weights for each criterion and assist the DM in making decisions.

Because the number of pair-wise comparisons grows exponentially with the number of criteria, Saaty and Ozdemir (2003) suggest an upper bound limit of 7 ± 2 on the number of alternatives. Surveying the literature on papers applying AHP for the supplier selection, Ishizaka et al. (2012) have indeed verified that when the number of alternatives is high AHP becomes inappropriate.

The so-called ELECTRE type methods are also widely accepted in the literature, and are an interesting alternative to AHP, especially when the number of alternatives is higher than seven. Simos (1989, 1990) proposed a very simple procedure using a set of cards, allowing any DM to express her/his opinions on the relative importance of criteria, and to derive weights for each criterion. Later, Figueira and Roy (2002) proposed a new and refined version of this procedure which produces more consistent weights. For the sake of simplicity, from now on let us refer to this procedure as the *revised Simos' procedure* (RSP).

In this chapter, we study a practical case found in a large Italian company called H2H Facility Solutions S.p.A., with headquarters in Bologna (Italy). The company provides facility management solutions for hundreds of clients and more than 7000 buildings in Italy. They are responsible for performing periodical maintenance concerning electrical, mechanical, security, safety, cleaning and many other services. To this end, the company uses mostly third-party service providers (suppliers) to perform services according to customers needs. They are in contact with thousands of suppliers in Italy, therefore, it is fundamentally important for the company to be able to efficiently and reliably select the most suitable suppliers to provide services to their clients.

Our objective is to develop a *decision support system* (DSS) to aid their DMs in selecting suitable suppliers for each client contract. In this sense, we present the results of the first phase of the project, where we define the criteria used by DMs to select suppliers and use two approaches to derive weights for these criteria. In a second phase, these weights are going to be refined and validated in order to be later used by the DSS to rate suppliers and propose suggestions to DMs. This project is also an interesting application of the AHP, given that this method is usually applied in supplier selection only with a reduced number of alternatives, while in our case there are thousands of them. It is also worth mentioning that the methodology proposed in this work can be extended to many other problems in the context of decision making in global service providers.

The remainder of the chapter is organized as follows. In Section A.2 the details of the

case study are presented. Then, in Section A.3, the results of the weight evaluation for both the AHP and the RSP are shown and in Section A.4 some conclusions are drawn and details on future research are provided.

A.2 Case study

When a new client approaches or is approached by H2H, it usually requires the company to manage only certain types of services in its facilities. Each type of service requested by a client is considered as a separate contract. Then, the DMs at H2H must select the most suitable supplier for each contract. Once the contracts are established, the associated suppliers may be required to perform not only scheduled services but also others that are requested only when needed.

It is often the case that a client requests eventual *extra* services that are not covered by their contracts, and thus are paid apart. In such a case, the company first requests the supplier which has the contract with the client to perform the extra service. If the supplier refuses there are two options. The first is to request the service from a supplier that has a special type of contract with H2H, where it is not bound to any specific client, but instead it is available to perform only extra services. Alternatively, the company can opt to pay other suppliers per extra service performed without any contracts.

Currently, the DMs at the company have to rely mostly on experience and knowledge of the market to assess the best supplier for each contract. As a consequence, the analysis becomes somehow subjective and DMs have a tendency to concentrate on a reduced pool of suppliers to contact. In this sense, the main aim of the project is to develop a DSS that will enable DMs to have a preliminary automatic analysis on the most suitable suppliers for a given contract and then, based on the results, use their experience to make their choice. This approach improves their ability to deal with an extended pool of options, without undermining the importance of the DMs experience in the final decisions.

In order to better understand which criteria are used by DMs when choosing a supplier we performed a series of interviews with several employees from different sectors of the company. From these interviews we were able to list the main criteria and build a 3-level hierarchy. In the first level there are the so-called *macro criteria*, while the second is composed by *micro criteria* and the third by *nano criteria*. In the following we present each macro criterion and their associated micro criteria. Although we provide a description on the macro criteria, for the sake of confidentiality we omit details on the micro and nano criteria.

Economic indicators (ECI): This macro criteria evaluates the economical stability of each supplier, and it is composed by the following two micro criteria: Annual revenue (REV); and level of debt (LDE).

Technical and professional capabilities (TPC): Evaluates the organizational structure,

skills and competences of the supplier. This macro criterion is composed by the following 5 sub-criteria: Number of workers per number of service types offered by the supplier (WPS); number of office employees per total number of employees (OPE); number of qualification certificates provided per number of employees (QPE); annual revenue per number of employees (RPE); and number of cities in which the supplier is able to provide services per number of branches (RPH).

Level of saturation (LSA): This macro criterion embeds measurements that indicate the level of saturation of a supplier, providing insights on whether or not it is still capable of accepting more contracts. This criterion might also provide evidence on an excessive dependence of the supplier with respect to the company. It is subdivided into the following micro criteria: Number of contracts per number of workers (CPW); aggregated area from assigned contracts to the supplier per number of workers (APW); and annual revenue generated by contracts with H2H per total annual revenue of the supplier (RPR).

References (REF): new suppliers are required to provide contact information of clients for which they have previously worked with. The company then might contact these clients to evaluate the quality of the services performed. This macro criterion is divided into the following sub-criteria: Number of references (NRF); and average score of references (SRF).

Level of service performance (LSP): Refers to aspects related to the quality of service performance, flexibility of the supplier and feedbacks from both the client and H2H. It is composed by the following micro criteria: Operational punctuality (OPT); administrative punctuality (APT); flexibility (FLX); quality (QLT); feedback H2H (FEH); and feedback client (FEC).

Furthermore, some of the aforementioned micro criterion of the macro LSP are further divided into nano criteria as follows.

Operational punctuality (OPT): Percentage of overdue service requests with respect to the total requests generated from the supplier contracts (OTC); percentage of overdue service requests with respect to the total of requests from extra services (OTE); percentage of budget reports presented in delay (BRD); and percentage of service requests performed in delay that were originated from budget reports (TPD).

Administrative punctuality (APT): Percentage of qualification documents delivered in delay (QDD); and percentage of reports not filled correctly (RFC).

Flexibility (FLX): Percentage of extra services performed with respect to the number of services performed that are only related to the contracts of the supplier (EPC); percentage of extra services refuted by the supplier (ESR); and number of service requests not performed (SNP).

Quality (QLT): Percentage of extra service requests proposed by the supplier with respect to the total of extra service requests performed (ESP); percentage of accepted budget reports with respect to the total number of such reports that are presented by the supplier

(ABR); percentage of anomalies encountered by DMs with respect to the total number of service requests performed by the supplier (ANT); percentage of service requests in which another supplier was required (CSS); percentage of anomalies signaled by suppliers through the smartphone app of the company (ANP); and percentage of service requests generated from contracts that were not performed by the supplier (TNP).

Feedback H2H (FEH): Average score given by DMs for the supplier (ASS); and index of economy (INE).

A.3 Implementation and results

To apply the AHP in this context, we first created an online survey in which 20 H2H employees from different sectors were requested to compare the relative importance of each criterion in a pair-wise fashion. For each comparison they were given a set of options as depicted in Figure A.1. For respondents it is easier and more intuitive to judge the importance of criteria with verbal judgments instead of with numerical ones. Notice that only criteria of the same hierarchy level are compared at a time. For example, we first perform a pair-wise comparison between the macro criteria, then between micro criteria of ECI, and so on.

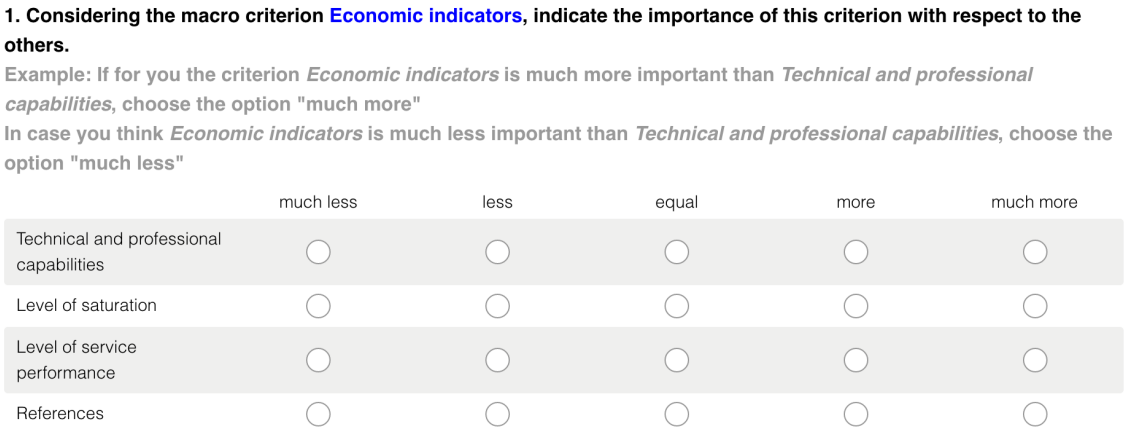


Figure A.1: Example of pair-wise comparison in the survey.

In the AHP, to derive the weights for each criterion we must first convert verbal judgments into numerical ones. In the literature, there are a variety of scales for comparing alternatives (see, e.g., Ishizaka2012). In this sense, we opted to use the linear scale proposed by Saaty (1977), which is perhaps the most commonly used. In this scale judgments are associated with integers from 1 to 9. In our case, we considered the following association of alternatives to numerical values.

Basically, there are two main approaches to analyze group decisions in the AHP. In the first one, called *aggregation of individual judgments* (AIJ), before computing the array of priorities (weights), individual judgments are aggregated as if the group suddenly became

Table A.1: Association of verbal judgments to a numerical scale.

much less	less	equal	more	much more
5	3	1	3	5

a single individual. There is a significant discussion in the literature about whether or not this approach violates the *Pareto principle*, which states that: "given two alternatives A and B, if each member of a group of individuals prefers A to B, then the group must prefer A to B" (Forman and Peniwati 1998). According to Forman and Peniwati (1998), in case the group is acting in concert and we are not concerned with individual priorities the Pareto principle becomes irrelevant and this approach might be used. In the second approach, called *aggregation of individual priorities* (AIP), we first compute the array of priorities for each individual and then, using either *arithmetic* or *geometric* mean, we aggregate priorities into a single array, which defines the final weights for criteria. Forman and Peniwati (1998) shows that neither method of evaluating the mean violates the Pareto principle in the AIP, but recommends using geometric mean to derive more consistent weights. Because we are indeed concerned with individual priorities, we decided to use AIP and aggregate the priorities with geometric mean. Furthermore, note that our AHP approach was implemented in C++ and no commercial software was used. In Figure A.2 we present the resulting weights. Notice that among the macro criteria, TPC and LSP seem to be the most important. This is actually not a surprise because these criteria are related to the capabilities of the supplier and the quality of services performance, which are expected to be very important factors. In general, from the analysis of the AHP results, it is possible that some criteria have a very small weights and are almost irrelevant, however this was not the case and all criteria evaluated seem to be relevant.

As a complementary analysis, we applied the RSP method, which is quite a dynamic method and requires respondents to sort the criteria in order of importance using a set of cards, where each card represents a single criterion. In addition, there are some *white cards* that serve the purpose of specifying a larger difference between criteria. Finally, respondents are required to inform a numerical value z defining the difference between the least important criterion with respect to the most important. To illustrate, consider the example depicted in Figure A.3, showing the order in which the criteria $\{A, B, C, D, E, F\}$ have been sorted. The presence of the white card in the second level means that there is a larger difference between C and B than to B and A. Notice that, in this example $z = 4$, thus criteria in the first level (i.e., C, E and F) are four times less important than criteria in the last level (i.e., A and D).

We then performed a second round of interviews to ask employees to sort the criteria. For each respondent, the procedure RSP was used to derive the weights for criteria, then individual weights were aggregated using geometric mean. As for the AHP, the algorithm was implemented in C++ and the resulting final weights are depicted in Figure A.4. Notice

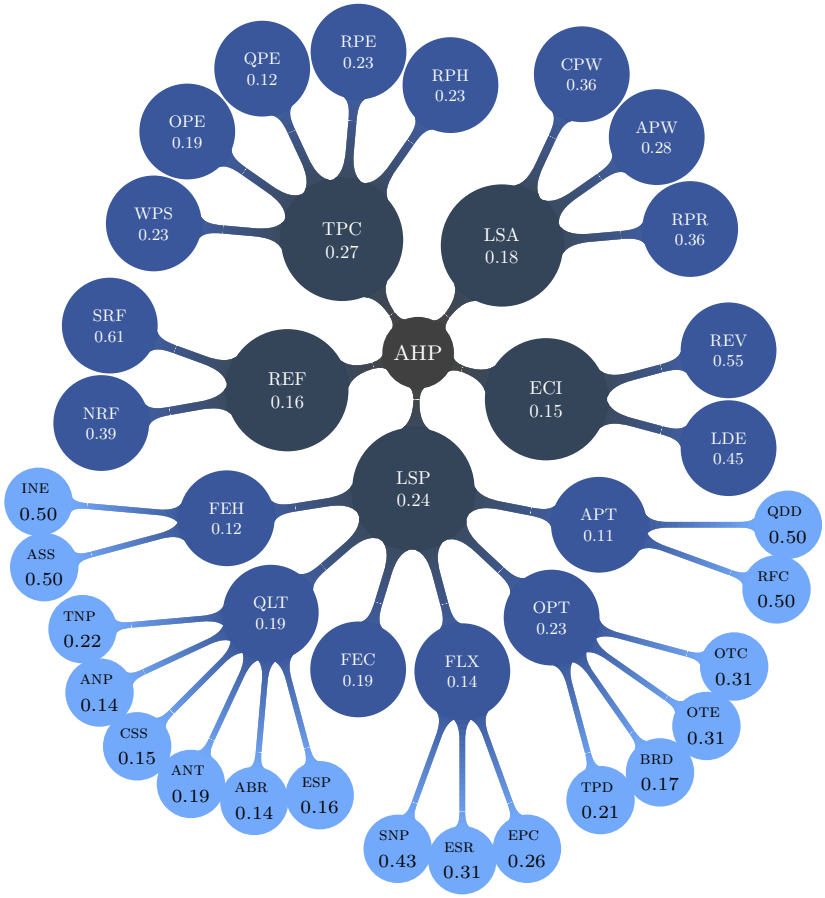


Figure A.2: Resulting weights derived from the AHP and aggregating weights with the AIP approach with geometric mean.

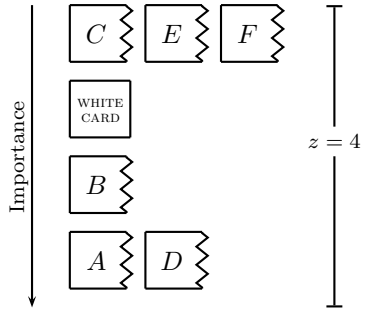


Figure A.3: Example of the sorting step required for the RSP.

that, although in this scenario the macro LSP was rated slightly more important than TPC, these criteria are still the most important. In general, apart from small differences these results are very consistent with the ones from the AHP.

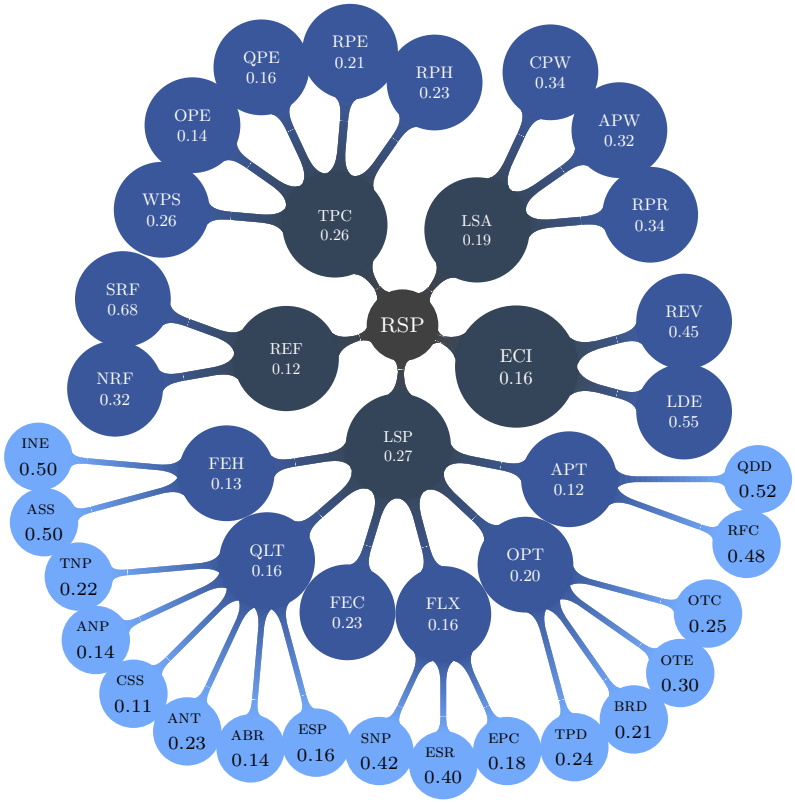


Figure A.4: Resulting weights derived from the RSP, aggregating weights with geometric mean.

A.4 Conclusions and future work direction

In this chapter, we presented the results of the first phase of a project to develop a DSS for a large Italian company. By interviewing employees we managed to list the main criteria used by DMs to choose the most suitable suppliers. Then, by surveying 20 employees from different sectors, we extracted their personal opinions regarding the relative importance of the criteria and used the well known AHP method to derive weights for each criterion.

Later, we performed a second round of interviews and gathered data for applying the RSP method, which also enables us to derive weights for criteria. The results of both methods are very similar, providing us with initial evidences of satisfactory consistency.

As future research, we plan to apply a range of *business intelligence* techniques (see, e.g., Vercellis 2009) for the simulation of the DSS, using past data from the company and the proposed weights. In addition, a mathematical model could be used to provide an approximation of the optimal assignments of suppliers to contracts. The results of both simulations could be compared with each other in order to validate the correctness and consistency of the proposed weights. This could enables us to refine the weights and implement the first prototype of the DSS to be deployed at the company and tested.

A.5 Bibliography

M. Celik, A. Kandakoglu, and I.D. Er. Structuring fuzzy integrated multi-stages evaluation model on academic personnel recruitment in MET institutions. *Expert Systems with Applications*, 36(3 PART 2):6918–6927, 2009.

I. Chamodrakas, D. Batis, and D. Martakos. Supplier selection in electronic marketplaces using satisficing and fuzzy AHP. *Expert Systems with Applications*, 37(1):490–498, 2010.

J. Figueira and B. Roy. Determining the weights of criteria in the ELECTRE type methods with a revised Simos' procedure. *European Journal of Operational Research*, 139(2):317–326, jun 2002.

E. Forman and K. Peniwati. Aggregating individual judgments and priorities with the analytic hierarchy process. *European Journal of Operational Research*, 108(1):165–169, 1998.

A. Ishizaka and A. Labib. Analytic Hierarchy Process and Expert Choice: Benefits and limitations. *OR Insight*, 22(4):201–220, 2009.

A. Ishizaka and A. Labib. Review of the main developments in the analytic hierarchy process. *Expert Systems with Applications*, 38(11):14336–14345, 2011.

A. Ishizaka, C. Pearman, and P. Nemery. AHPSort: an AHP-based method for sorting problems. *International Journal of Production Research*, 50(17):4767–4784, 2012.

R. Khosla, T. Goonesekera, and M.T. Chu. Separating the wheat from the chaff: An intelligent sales recruitment and benchmarking system. *Expert Systems with Applications*, 36(2 PART 2):3017–3027, 2009.

A.W. Labib. A supplier selection model: a comparison of fuzzy logic and the analytic hierarchy process. *International Journal of Production Research*, 49(21):6287–6299, 2011.

S.-H. Lee. Using fuzzy AHP to develop intellectual capital evaluation model for assessing their performance contribution in a university. *Expert Systems with Applications*, 37(7): 4941–4947, 2010.

T.L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15(3):234–281, 1977.

T.L. Saaty and M.S. Ozdemir. Why the magic number seven plus or minus two. *Mathematical and Computer Modelling*, 38(3-4):233–244, 2003.

J Simos. *Lévaluation environnementale: un processus cognitif négocié. 216 p.* PhD thesis, Thèse 204 Thèse de doctorat/2011-Emilie Chardine-Baumann, 1989.

J. Simos. Evaluer l'impact sur l'environnement: Une approche originale par l'analyse multicritère et la négociation. In *Evaluer l'impact sur l'environnement: une approche originale par l'analyse multicritère et la négociation*. Presses polytechniques et universitaires romandes, 1990.

C. Vercellis. *Business Intelligence: Data Mining and Optimization for Decision Making*. 2009.

L.A. Vidal, E. Sahin, N. Martelli, M. Berhoune, and B. Bonan. Applying AHP to select drugs to be produced by anticipation in a chemotherapy compounding unit. *Expert Systems with Applications*, 37(2):1528–1534, 2010.

T.Y. Wang and Y.H. Yang. A fuzzy model for supplier selection in quantity discount environments. *Expert Systems with Applications*, 36(10):12179–12187, 2009.