Tactical Problems in Vehicle Routing Applications

Francesco Bertoli

A thesis submitted for the degree of
Doctor of Philosophy of
The Australian National University

July 2018

Except where otherwise indicated, this thesis is my own original work.

Francesco Bertoli
18 July 2018

To my beloved little mushroom, my chick.

Acknowledgments

I first wish to thank my primary supervisor: Phil. I guess it's not easy to take in a Ph.D. student half way through his time. Especially if he moves to Sydney before starting to work with you. Thanks for the time you dedicated to me. In particular, I want to thank you for helping me find my way in the research world. Before turning my attention to OR, I was losing my interest in research.

A huge "thank you" goes to Tommaso. For many many reasons. First, because you helped me in my transition. Then for all your attention and dedication, and for being a real-time available library on programming. Last but not least, for hosting me pretty much every time I came to Canberra and for making those visits fun. I really enjoyed working together, and I hope we'll have another chance.
I also wish to thank Adrian. If I came to Australia for my Ph.D., it was thanks to you. Moreover, I appreciated you following me every day while we were working together. I also appreciate the opportunity you gave me to visit and give seminars in several cool places.

I must, and want to, give thanks to my family. My parents, for being my parents and a limitless source of support. My brother, for being my brother. I cannot explain it a different way. Thanks for coming down to the other side of the world several times. It meant a lot and it made the distance shorter.

There are so many people that I have met during these four years that I want to remember and acknowledge. Enrico, Sara and Alessandro, for giving me an (Italian) place to stay in Canberra. I hope I'll see you again back home. Antonio, Pierrick, Hannah and their cool housemates, and Andrei and Arthur, for their hospitality, which I appreciated very much. I had much fun staying with you guys. I really hope to meet again somewhere in the world. Alessandro, for making me discover the OR world. The Couchsurfing community in Canberra, you made it easy to move to a new continent. I don't know what I would have done without the Tuesday meetings. The Awesome Kids, which I think is the nicest group of people I have ever met. Remy, Laura, David, Jackie, Mark for all the fun climbing together. Nick and Tracy, I can now say thanks for the trust. The Sydney crew, no need to say we will see each other

again. Finally, thanks to the Hivery team, for making me living a great experience.

I saved the most important person that I have had during this period for last: Marghe. I am so thankful you came here and I had you on my side. I don't even know how we made it through a year and a half of intercontinental relationship but I am so grateful we did. I am so excited that you will be with me in my future, in our future. My Ph.D. time is truly blended with my time with you. It does not make sense to consider them as two separated things. If I had to thank a person, that'd be you. I love you.

Abstract

The class of Vehicle Routing Problems (VRPs) is one the most studied topics in the Operations Research community. The vast majority of the published papers focus on single-period problems, with a few branches of the literature considering multi-period generalisations. All of these problems though, consider a short horizon and aim at optimising the decisions at an operational level, i.e. that will have to be taken in the near future. One step above are tactical problems, i.e. problems concerning a longer time horizon. Tactical problems are of a fundamental importance as they directly influence the daily operations, and therefore a part of the incurred costs, for a long time. The main focus of this thesis is to study tactical problems arising in routing applications.

The first problem considered concerns the design of a fleet of vehicles. Transportation providers often have to design a fleet that will be used for daily operations across a long-time span. Trucks used for transportation are very expensive to purchase, maintain or hire. On the other side, the composition of the fleet strongly influences the daily plans, and therefore costs such as fuel or drivers' wages. Balancing these two components is challenging, and optimisation models can lead to substantial savings or provide a useful basis for informed decisions

The second problem presented focuses on the use of a split deliveries policy in multi-period routing problems. It is known that the combined optimisation of delivery scheduling and routing can be very beneficial, and lead to significant reductions in costs. However, it also adds complexity to the model. The same is true when split deliveries are introduced. The problem studied considers the possibility of splitting the deliveries over different days. An analysis, both theoretical and numerical, of the impact of this approach on the overall cost is provided.

Finally, a districting problem for routing applications is considered. These types of problems typically arise when transportation providers wish to increase their service consistency. There are several reasons a company may wish to do so: to strengthen the customer-driver relationship, to increase drivers' familiarity with their service area, or, to simplify the management of the service area. A typical approach, considered here, is to divide the area under consideration in sectors that will be subsequently assigned to specific drivers. This type of problem is inherently of a multi-period and tactical nature. A new formulation is proposed, integrating standard routing models into the design of territories. This makes it possible to investigate

how operational constraints and other requirements, such as having a fair workload division amongst drivers, influence the effectiveness of the approach. An analysis of the cost of districting, in terms of increased routing cost and decreased routing flexibility, and of several operational constraints, is presented.

Glossary

TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
CVRP	Capacitated VRP
PVRP	Periodic Vehicle Routing Problem
IRP	Inventory Routing Problem
SDVRP	Split Deliveries VRP
MDSDVRP	Multi Day Split Deliveries VRP
FSM	Fleet Size and Mix
DC	Distribution Centre
TS	Tabu Search
SA	Simulated Annealing
LNS	Large Neighborhood Search
ALNS	Adaptive Large Neighborhood Search
CG	Column Generation
TDP	Territory Design Problems
TBR	Territory Based Routing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we briefly illustrate the domain to which the problems studied belong. Moreover, we outline the objectives and contributions of this thesis. Finally, we present the structure of the following chapters.

1.1 Background and Motivation

One of the most active areas of Operations Research (OR) is certainly the family of Vehicle Routing Problems (VRPs). In their seminal work, Dantzig and Ramser [1959], published more than 50 years ago, Dantzig and Ramser proposed the first mathematical programming model and the first numerical method for the VRP. Since then a huge number of papers have been published, proposing new variants of the original problem, new models, exact methods and heuristic algorithms. The interest of the research community is not only motivated by the fact that the VRP constitutes a hard and challenging combinatorial problem, but also by the applicability and industrial relevance of the problem. The use of optimisation techniques in the routing industry is indeed regarded as one of the success stories of OR. It is not only due to the advent of modern computers, but also, and in particular, to the mathematical understanding of the VRPs and to the complexity and efficient implementations of all the optimisation techniques that have been proposed and studied in the literature.

In the past years, the focus of the research community has partially moved away from academic extensions of the VRP and has shifted towards more realistic and industrial variants. These problems deal with multi-faceted objective functions, aimed at modelling various sources of cost, uncertainty and dynamism. A variety of complex operational constraints are studied, aiming to better represent the needs of transportation providers. These problems usually go under the name of rich VRPs. There are many papers attempting to survey new trends and models in the routing literature, two examples are Caceres-Cruz et al. [2015] and Lahyani et al. [2015]. An interesting paper comparing the state of the art of scientific research and commercial

software used to solve real-world VRPs is Drexl [2012a]. In this work, the author also points out the gaps that still separate the literature from real-world applications.

The VRP, as well as most of its variants, is by nature a single-period problem, i.e. it considers only one period (usually a day) of operations. However, there are areas of the literature focusing on problems featuring a longer planning horizon (multi-period). Two of the most know examples are the Inventory Routing Problem (IRP, Coelho et al. [2013]; Andersson et al. [2010]; Bertazzi and Speranza [2012]) and the Periodic Vehicle Routing Problem (PVRP, Francis et al. [2008]; Irnich et al. [2014a]). These two problems offer two common examples of reasons to consider a multi-period horizon and integrate the routing component, i.e. how to deliver on each period, to the scheduling component, i.e., when to deliver. On one side, the IRP attempts to combine different phases of the supply chain, such as inventory control and delivery. On the other side, the PVRP models applications where there is strong periodic component in the daily delivery operations. In both cases, integrating the scheduling and routing parts of the problem can lead to substantial savings (Andersson et al. [2010]; Francis et al. [2008]). However, the research concerning multi-period problems is under-developed when compared to single-period VRPs. As an example, even though the IRP is probably the most studied multi-period routing problem, there is no shared abstract model (Andersson et al. [2010]). Moreover, given the computational difficulty of the problem, only recently exact algorithms were introduced. Although, there are only able to solve very small instances. In fact, most methods are heuristic (Coelho et al. [2013]).

Even though there are multi-period extensions of the VRP, the vast majority of the problems focus on optimising decisions at an *operational* level. According to Crainic [2003]; Kilby and Urli [2016]; Hoff et al. [2010], in transportation systems there are three main management levels, hierarchically ordered: strategic, tactical and operational. The main difference can be identified in the time horizon considered in each level. The strategic is the highest level of management, and typically involves large capital investments over long-term horizons. Strategic decisions concern development policies and broad determination of strategies. Tactical level decisions aim at planning, over a medium-term horizon, an efficient allocation and utilisation of resources. Although in routing applications, this may include the determination of the routes or exploit the available routing information, the main problem solved is not the routing problem. Instead it is to decide what, and partially how, resources will be used throughout the horizon. At the operational level only short time-horizon decisions are considered. A key point of this classification is that decisions at each level constrain the decisions at the following level, and inform the decisions at the previous level.

The vast majority of papers in the routing literature are dedicated to problems at the operational level, with fewer papers studying how routing models can be extended to tackle tactical problems. However, tactical decisions have a strong influence on daily operations. As an example, the object of a chapter in this thesis is the design of a fleet of vehicles. This usually requires a significant investment and is not an operation that can be done daily, but instead has to consider a long horizon. The decisions taken will affect the future operational plans over a long term. This type of problem has largely been overlooked in the literature.

1.2 Objectives and Contributions

This thesis focuses on tactical problems arising in routing applications. Each chapter in Part II is dedicated to a different problem. However, there are similarities in the main lines of research of each chapter that match the general objectives in this thesis. Namely, they are:

- investigate how standard routing models can be extended so that operational decisions can be integrated into models for tactical problems;
- propose efficient solution methods for the formulated model;
- analyse the impact of the considered approaches at the operational level;
- analyse and quantify the impact of different features of models and instances on the efficiency of such approaches.

Not all objectives are pursued in equal measure in all chapters. In the following we briefly summarise the main topics studied in each chapter of Part II, the findings and the relative publications.

Fleet Design. Chapter 4 is dedicated to fleet design problems. The chapter describes a general model for fleet design problems over a long-time horizon. The solution assumes the existence of an operational level (daily instance) solver, and leverages this through the use of mathematical programming and the Dantzig-Wolfe decomposition. The method presented is thus very general. The contributions of the chapter are twofold: first, to present the first broad study on fleet design, along with a general model supporting several features that make the model applicable to real-world problems; second, to present decomposition methods that allow us to leverage existing solution methods for the class of single-period routing problems, making our proposed method extremely general and applicable. The chapter is based on the following papers:

- Francesco Bertoli, Philip Kilby and Tommaso Urli. **A general and scalable column generation approach to fleet design problems**. *Submitted to Journal of Heuristics*. 2017.
- Francesco Bertoli, Philip Kilby and Tommaso Urli. **A Column Generation-Based Approach to Fleet Design Problems Mixing Owned and Hired Vehicles**. *Submitted to International Transactions in Operational Research*. 2017.

The methods proposed in the chapter were object of a (pending) patent application in Australia and the United States. The references are:

- Australian Patent Application No. 2017203827
- US Patent Application No. 15/615,054

Splitting Deliveries in Routing Strategy. Chapter 5 considers the possibility of splitting deliveries in a multi-period VRP. Unlike split delivery routing as usually considered in VRP settings, we allow deliveries to be split across vehicles and across days. The main contribution is to analyse the benefits of this approach. A new problem is formulated, together with a mathematical programming model and some valid inequalities to enhance it. Additionally, a simple yet effective heuristic is proposed. Finally, a theoretical and numerical analysis is presented, aimed at quantifying the savings introduced by allowing split deliveries over days, and studying the impact that different features of instances have on the possible savings. The chapter is based on the following paper:

- Francesco Bertoli, Philip Kilby and Tommaso Urli. **Vehicle routing problems with deliveries split over days**. *Journal on Vehicle Routing Algorithms*. 2017.

Territory Design. The subject of Chapter 6 is the problem of dividing the service area for a transportation company. This falls in the category of districting problems. The chapter describes the proposed approach to tackle such problems for routing applications. The novelty lies in the integration of the routing decisions in the design model and solution method. This brings several advantages making it possible to consider operational constraints and balance requirements that are seldom considered in the literature on territory design. Another contribution is the extensive computational analysis presented. This aims at studying the effectiveness of the approach and quantifying the impact that different factors have on its effectiveness. The chapter is based on the following paper:

- Francesco Bertoli and Philip Kilby. **Territory Design For Routing Problems**. *To be submitted to Transportation Research: Part B*. 2017.

Other Contributions Here, we mention other contributions relative to a different research project studied during the first half of my PhD. The topics of this project were the theoretical analysis of Monte Carlo methods and their application to Stochastic Optimal Control Problems. The two main problems considered were the analysis of localisation operations in order to reduce the errors due to the system dimension in Particle Filtering methods, and the application of Monte Carlo methods to the estimation of optimal control in stochastic systems. The following papers were produced during the time spent on this project:

- Francesco Bertoli and Adrian N. Bishop. **Reducing the Bias in Blocked Particle Filtering for High-Dimensional Systems**. *Available at* https://arxiv.org/abs/1407.0220. 2014.
- Francesco Bertoli and Adrian N. Bishop. **Adaptively Blocked Particle Filtering with Spatial Smoothing in Large-Scale Dynamic Random Fields**. *Available at* https://arxiv.org/abs/1406.0136. 2014.
- Francesco Bertoli and Adrian N. Bishop. **Nonlinear Stochastic Receding Horizon Control: Stability, Robustness and Monte Carlo Methods for Control Approximation**. *International Journal of Control. 2017. Url:* https://doi.org/10.1080/00207179.2017.1349340
- Francesco Bertoli and Adrian N. Bishop. **Monte Carlo methods for controller approximation and stabilization in nonlinear stochastic optimal control**. *Invited Paper at 17th IFAC Symposium on System Identification*. 2015
- Francesco Bertoli and Adrian N. Bishop. **An Error Analysis in the Limit Approximation in Path Integral Control**. *To be submitted.*

1.3 Uncertainty in Tactical Problems

A key point in the thesis is the way uncertainty is handled. As mentioned in Section 1.1, strategic, tactical and operational levels differ in terms of the time horizon considered. It is clear though, that the longer the horizon is, the higher the uncertainty of the problem will be. At the operational level, uncertainty is often not considered. For example, in the VRP the input is completely known. However, in the vehicle routing literature there are variants of the VRP, considering, for instance, stochastic travelling times (Malandraki and Daskin [1992]) or stochastic demands (Bertsimas [1992]). We refer to Gendreau et al. [2014] for a survey of the stochastic variants of the VRP. At the strategic level, uncertainty must be considered as it plays

an important and non-negligible role (Crainic [2003]). At the tactical level, uncertainty is still present and should not be neglected (Hoff et al. [2010]). Uncertainty is handled in different ways, depending on the problem studied. In each chapter, while we review the relevant literature, we review some of the approaches proposed in the literature that consider uncertainty. The approach we take in this thesis is to use historical data, or any other set of scenarios that is able to capture day-to-day variation in customers and demand, as the input to a deterministic model. This is similar to a scenario approach, where different realisations of the single operational days are considered. This choice is motivated by several reasons. First, it is a successful and powerful approach. An example is given in Kilby and Urli [2016], where the authors show how well this approach performs on a fleet design problem. Second, but very important, it allows the design of efficient methods and algorithms. One of the reasons tactical problems have been overlooked in the literature is the high level of uncertainty (and the resulting huge size of a full stochastic model). Modelling uncertainty directly can lead to a very big and challenging problem. We attempt to use (deterministic) operational models to build models and methods for tactical problems. This allows us to leverage the efficiency and power of that the methods for operational problems have reached. One last reason is the availability of data. Many stochastic approaches make use of probability distributions in order to describe the problem. These are used directly or only for sampling purposes. If a probability distribution is not given, it must be inferred from the available data. However, we need a certain amount of data if we wish to extrapolate a reliable distribution and not always we have access to that much data. As an example, in Chapter 6, we only have a week worth of data. Attempting to fit a distribution on such a small data-set can lead to a very unreliable input in our problem, which in turn would lead to poor quality solutions. In these cases, using the available data set as input to our problem is the only viable approach. On the other hand, if we do have a probability distribution available, we can use it to generate a reasonable number of scenarios as input to our problem. In each chapter, we highlight the advantages of our approach and the reasons to choose it over other approaches.

1.4 Thesis Outline

The thesis is organised as follows: in Part I, the necessary background material is reviewed. In Chapter 2, we present a general overview of routing problems. We focus specifically on the areas relevant to this thesis. An overview of the solution methods that are used in the thesis can be found in Chapter 3. Part II is divided in three chapters, one for each of the main problems considered in the thesis, as presented

above. In each chapter, we present the motivation for the research carried out, a review of the relevant literature, the contributions to the field and some conclusions.

1.5 Notation

The problems considered in Part II are all multi-period extensions of the VRP. For the sake of clarity, we use the same notation throughout the entire thesis. Here, we introduce the main concepts, symbols and naming conventions that are recurrent in the following chapters.

- The number of customers considered is denoted by N and the set of customers by $C = \{1, \ldots, N\}$.
- The depot is denoted by 0, and the set of all nodes in the network is denoted by $\overline{C} = \{0, 1, \ldots, N\}$.
- Generally, customers are denoted using letters i or j.
- The cost of travelling between two nodes $i, j \in \overline{C}$ is denoted by c_{ij}.
- The time window of a customer $i \in C$, is denoted by $[e_i, l_i]$.
- The fleet, or set of vehicles, is denoted by $F = \{1, \ldots, |F|\}$.
- The capacity of a vehicle $v \in F$ is denoted by Q_v. In case the fleet is homogeneous, we only write Q.
- The set of periods (single unit of operations, e.g., days) is denoted by D.
- Periods (i.e., days) are denoted using the letter d.
- The demand of a customer $i \in C$ on period $d \in D$ is denoted by q_{id}. We write q_i for single-period problems.
- The set of routes for a routing problem is denoted with R. Super and subscripts are used to denote dependence on other parameters, e.g., R_d is the set of routes on day $d \in D$.

Part I

Mathematical Background

Chapter 2

Routing Problems

The goal of this chapter is to cover some of the background for Part II. In particular, an overview of routing problems is presented, with a particular focus on the areas relevant to this thesis.

2.1 The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one of the most studied problems in the Operations Research community. In its simplest form the VRP is a straightforward generalisation of the Travelling Salesman Problem (TSP) and can be stated as follows. We are given a set of customers $C = \{1, \ldots, N\}$, a depot, denoted by 0, where a fleet of identical vehicles $F = \{1, \ldots, |F|\}$ is located, and the cost c_{ij} of travelling from location i to location j for all pairs of locations in $\overline{C} = C \cup \{0\}$. The goal is to find a set of routes, i.e., an ordered list of customers starting and ending at the depot, so that all customers are visited by a route and the total distance travelled is minimised. Arguably, the most known version of the VRP is the Capacitated VRP (CVRP). In this version, each customer $i \in C$ has a demand $q_i \geq 0$, representing the amount of goods to be delivered, and all trucks have a limited capacity $Q > 0$. A route is feasible only if the sum of the loads of customers visited by a route does not exceed the capacity. Here and in all the following chapters, we assume that the triangular inequality for the costs c_{ij} is valid.

The VRP was originally introduced in Dantzig and Ramser [1959], where a first mathematical programming formulation and solution method were presented. The VRP is known to be a $\mathcal{NP}$-hard problem (Karp [1972]).

In next sections, we briefly present some of the most common extensions and variants of the classic VRP. The presentation is by no means exhaustive. For a more comprehensive survey we refer to the paper by Laporte (Laporte [2009]) and the book by Toth and Vigo (Toth and Vigo [2014]).

2.2 Routing Problems

The classification of routing problems is a task hard enough to have several survey papers and books dedicated to it. In this section, we present some extensions and characteristics of particular interest in this thesis. There are many features and extensions hereby not considered, such as stochastic and dynamic extension, for which we refer to the surveys in Laporte [2009]; Caceres-Cruz et al. [2015]; Hasle and Kloster [2007] and the book Toth and Vigo [2014].

2.2.1 Requests

In the CVRP the requests are described as demands for goods that have to be satisfied by a route. Here we look at some requests' features that arise in different routing problems.

Time Windows. Amongst the most studied and practically relevant constraints associated with requests are those related to time. In the VRP with Time Windows (VRPTW, Cordeau and Groupe d'études et de recherche en analyse des décisions [Montréal(2000]) the time t_{ij} needed to travel between pairs of locations $i, j \in \overline{C}$, a time window $[e_i, l_i]$ and a service time s_i for each customer $i \in C$ are considered. Customers can only be visited during their time windows. Usually, vehicles are allowed to arrive early and wait at a location before the start of a time window. The service time represents the time needed to complete service at the customer's location. In some problems, multiple time windows for each customer are considered (Pesant et al. [1999]; Vidal et al. [2014]).

Time windows are not always imposed as *hard* constraints. In Taillard et al. [1997], the authors introduce the VRP with Soft Time Windows (VRPSTW) where early and late arrivals are allowed but penalised by using a convex function. The most common penalty functions used to penalise time windows violations are piece-wise linear functions. A mix of penalty functions and/or bounds on late service or early arrivals is also possible.

Pairing Precedence. Request do not have to be independent from each other. An important example is the class of Pickup and Delivery Problems (PDPs, Parragh et al. [2008, 2007]). In these problems, goods, or passengers, have to be transported from some origin locations to different destinations. These problems are usually classified according to how the pickup and delivery locations relate to each other (Parragh et al. [2008]). Arguably, most of the papers in the literature focus on problem where each request consists in transporting a single commodity from one location (pickup)

to another one (delivery). If no transhipments are allowed, then the same vehicle must visit both locations and the pickup must be performed before the delivery.

Split Deliveries. In the CVRP, requests can only be performed by a single vehicle. However, there are many reasons to consider the possibility of splitting the service of a customer across multiple vehicles. One simple example is if some demands exceed the capacity of the biggest truck. In Dror and Trudeau [1989], the authors introduce the Split Delivery VRP (SDVRP). This is a straightforward extension of the CVRP where it is no longer assumed that a request has to be satisfied by a single vehicle. Allowing a delivery to be split adds a considerable complexity to the problem but can also lead to considerable savings, up to 50% (Archetti et al. [2006a]).

Profitable Routes. It is not possible to satisfy all requests in all applications, due to capacity and time limitations. In order to select the requests to be fulfilled, one can penalise the unassigned requests or reward the ones that are included. The literature on routing problems with profits is vast. The academic formulations go from simple extensions of the TSP (Tsiligirides [1984]) to extensions of the VRP (Butt and Cavalier [1994]). A survey can be found in Archetti et al. [2014b].

Multi-Commodity Routing. One somewhat overlooked class of problems are those where customers' demands are made of several commodities. Some authors have proposed academic extensions of the VRP (Archetti et al. [2014a, 2015a]) to include more than one commodity. Multi-commodity routing applications are common in practice: two examples, which are the problems that motivated the work presented in Chapter 4, are: the delivery of fuel to gas stations and the delivery of grocery to supermarkets. In the former, the different types of fuel, that may not be mixed, are the commodities. In the latter, one has to differentiate between products that need, or do not need, to be transported in a refrigerated truck.

2.2.2 Routes

An important aspect in the formulation of a routing problem is the definition of a feasible route. This depends on the structure and constraints we impose. In the CVRP, the only constraint considered is given by the capacity of the trucks. Route related constraints can be divided into intra-route and inter-route classes. Inter-route constraints operate between routes, one example is the synchronisation of some task, for instance in technicians routing where more than one technician is needed to perform a job. Here we focus mainly on intra-route constraints: constraints that

operate *within* a route. A more comprehensive list of route constraints and structures can be found in Irnich et al. [2014b]; Drexl [2012a].

Resources. Simple constraints can be imposed on a route's length and time. These could be due to working regulations or a need to limit the consumption of some resources (see Laporte et al. [1984] for an example). Obviously, capacity can be considered as another limited resource.

Depot Visits. Another assumption in the CVRP is that a vehicle cannot return to the depot while performing a route. However, this assumption is often not true in practice, as vehicles are allowed to visit the depot to refill or unload. In Taillard et al. [1996], the authors introduce the VRP with Multiple use of vehicles to model and study this situation. This problem is also known as the Multi-Trip VRP (Cattaruzza et al. [2014]; Battarra et al. [2009]). An example of a real-world problem allowing for multiple trips is the problem studied in Chapter 4.

Balanced Routes. One example of an inter-route constraint, which has not received much attention, is the balance of the routes with a particular measure (Drexl [2012a]; Jozefowiez et al. [2008]). For example, a company may wish to have daily routing plans that are fair to the drivers in terms of time (Jozefowiez et al. [2007]). If a driver is paid proportionally to the load carried (Lee and Ueng [1999]), or the number of requests served (as in technicians routing) a fair division of the demand between the routes is essential. This matter is explored in more detail in Chapter 6.

Geometric Properties of Routes. In the majority of routing problems, the geometric properties of the routes are ignored. However, these may be of practical relevance in real-world applications. Some papers (Lu and Dessouky [2006]; Constantino et al. [2015]) have proposed different measure to make routes visually appealing. One other important concept is that of *regionalization*, as defined in Mourgaya and Vanderbeck [2007], of the routes. This is related with the goal of having drivers that are familiar with their service area. More references are given in Chapter 6.

2.2.3 Fleet

The available fleet and its characteristics are also of paramount importance in a routing problem. The term fleet might refer to the set of vehicles or the team of drivers. Here we only focus on features concerning the vehicles and refer to Drexl [2012a] for a fuller discussion on driver-related characteristics.

Heterogeneous Fleet. In the CVRP the vehicles are assumed to be identical. However, this is rarely the case in reality, as fleets are usually composed of different types of vehicles. The difference in the available vehicles might be reflected, for example, in different capacities and different travelling times and costs. Moreover, the fleet is not always given as an input; instead its composition can be a decision variable. If so, the cost of purchasing a vehicle can differ based on its type. A classification of the problems considering a heterogeneous fleet is given in Baldacci et al. [2008]. A more industrial oriented survey can be found in Hoff et al. [2010]. In Chapter 4, a more detailed survey on this topic is presented.

Compatibilities. Heterogeneous fleets give rise to compatibility constraints. Common reasons for incompatibility are access limitations, due to size or regulations, to a customer's location. These compatibilities may also be due to drivers' qualifications.

Multi-Compartment Vehicles. The counterpart of multi-commodity requests are multi-compartment vehicles. The number of compartments can be fixed, and each compartment can be dedicated to a single commodity (Mirzaei and Wøhlk [2017]) or several (Urli and Kilby [2017]). If the compartments take the form of trailers, there can be a variable number of trailers for each truck. This is the problem modelled by Chao in the Truck and Trailer Routing Problem (Chao [2002]). In Drexl [2013], the trailers can be exchanged between trucks during a route.

Multi-Depot. The vehicles might start and end at different depots. If the fleet is homogeneous, the problems is known as Multi Depot VRP (Renaud et al. [1996b]). Variation of this problems consider only the starting point of a vehicle while the end location is left open (Vidal et al. [2014]).

External Resources. Whenever the available fleet is not sufficient to meet all requests, a company might resort to hiring extra vehicles. The problem of sub-contracting some of the requests to external transportation providers appears in the literature in different areas. In Kopfer and Wang [2009] the authors discuss this problem, presenting different business options.

2.2.4 Objectives

The objective function may be composed of one or several components serving different goals, and can also be structured in different ways. Examples of single dimensional objectives are the minimisation of travelled distance or number of vehicles

used. If an objective is multi-dimensional, a hierarchical structure may be imposed. As an example, in the set of benchmark problems for the VRPTW presented in Solomon [1987], the first objective is to minimise the number of vehicles while the second is to minimise the total distance. If no hierarchy is assumed, the objective is a weighted sum of one dimensional objectives. Another source of cost are the constraints that are modelled as *soft* constraints. For example, a violation of the time windows may be allowed, but penalised, as discussed in (Taillard et al. [1997]). The survey presented in Jozefowiez et al. [2008] focuses on multi-objective optimisation in routing problems.

2.3 Multi-Period Problems

All of the VRP extensions presented in the previous section are of a single-period nature, i.e., the horizon considered is made up of only one period of operation. A part of the literature is dedicated to multi-period problems. Since periods are usually days, we use these two terms interchangeably in what follows. The reasons to introduce a multi-period horizon are several and of different nature. In some transportation systems, a single day might be not enough to describe the problem due to long distances (Savelsbergh and Song [2007]). One other reason is to integrate other aspects in the problem formulation. Two examples are provided by the well-known classes of problems denoted by Periodic VRPs (PVRPs) and Inventory Routing Problems (IRPs). In the former, the goal is to model a periodicity of the problem and integrate delivery-pattern schedule and routing. In the latter, the goal is to integrate inventory control and routing. Considering more than one period adds considerable complexity to the problems, but, on the other hand, offers the possibility for higher efficiency and cost savings. In this section, we consider some multi-period extensions of the VRP.

Periodic VRPs. In the PVRP a customer may request to be served one or more times in the planning period but the visit schedule is not fixed. The possible set of patterns of visits may be predefined for each customer. For example, a given customer could be visited twice: on Tuesday and Thursday, or on Monday and Wednesday. The PVRP and its extensions model applications with periodic delivery operations. An extensive survey on the PVRP and its extensions can be found in Francis et al. [2008]. For a recent review on exact methods and heuristic approaches we refer the reader to Toth and Vigo [2014]. Applications of practical relevance arise in grocery delivery (Carter et al. [1996]; Gaur and Fisher [2004]), waste collection (Angelelli et al. [2002]),

maintenance services (Hadjiconstantinou and Baldacci [1998]) and health care (Hemmelmayr et al. [2009]).

Inventory Routing Problems. In IRPs, there are no requests placed by customers. Instead, a vendor must decide when to visit each customer, how much to deliver and a routing plan for every day in the horizon (a business model called Vendor Managed Inventory). Usually a "consumption rate" for each customer is available and stock-outs are forbidden. The idea behind IRPs is to have a centralised control for both inventory and routing, integrating two parts of the supply chain that are usually controlled separately. This approach has specific applicability in real-world problems, as it requires complete control over the customers' inventory level, and also some form of knowledge of the rates of consumption. However, if such a deep integration of the inventory management and routing aspects is feasible, the possible savings are substantial (Andersson et al. [2010]). We refer to Coelho et al. [2013]; Bertazzi and Speranza [2012] for academic oriented survey on IRPs. An industrial point of view presented in Andersson et al. [2010].

Service Consistency. Another reason to consider a multi-period horizon is to increase the level of consistency in service. In Groër et al. [2009], the authors introduce the Consistent VRP (ConVRP) which is a multi-period routing extension of the VRP where the goal is to design daily routing plans so that customers are always visited by the same driver at approximately the same time. This problem was later extended in Kovacs et al. [2014] to have an assignment of teams of drivers to customers. Other examples of papers using multi-period routing models to improve service consistency are Smilowitz et al. [2013]; Schneider et al. [2014]; Wong and Beasley [1984]. The matter of improving consistency is analysed more in depth in Chapter 6.

Other Multi-Period Problems. Not all multi-period problems fall in the classes above. In Francis et al. [2008], the authors present the PVRP with service choice, where the delivery frequency impacts the customers' demand and service time. In Archetti et al. [2015b], the Multi Period VRP with Due dates is introduced. In this problem, one can choose the delivery date for each customer within a window of a few days. An inventory cost for holding the products in a central depot is considered. In Bertoli et al. [2017c], which is the base of Chapter 5, the possibility of splitting deliveries over consecutive days is considered and analysed.

2.4 Tactical Routing Problems

As mentioned in the Introduction, according to Crainic [2003], transportation systems management can be divided in three levels: *strategical, tactical and operational*. These three levels differ in the length of the horizon considered, from the longest (strategic level) to the shortest (operational level), and, consequently, also in the type of decisions taken, and the uncertainty faced. Examples of the decisions involved are: demand modelling and location of facilities for the strategic level, service network design and resource acquisition for the tactical level, routing and loading of vehicles for the operational level. In Crainic [2003], it is noted that decisions taken at a particular level influence lower levels and provide useful information to higher levels.

Tactical problems do not (only) focus on routing decisions, as these will likely be re-optimised and changed in the future, but also on issues that can be placed one step above in a decisional hierarchy. Some examples of tactical problems are considered and analysed in this thesis: the design of a fleet for a medium to long term or the districting of a service area for routing services such as grocery delivery or winter gritting operations. The literature on these topics is reviewed in the chapters of Part II. One other important example of tactical problems is the Location-Routing Problem (LRP, Nagy and Salhi [2007]. While facility location problems are more of a strategical nature (Crainic [2003]), the LRP can be described as follows: given a set of potential depots, costs for opening such depots, a homogeneous fleet and a set of customers with known demands, identify a subset of depots to be opened, assign customers to them, and determine vehicle routes so to minimise the total cost. A survey on LRPs is Prodhon and Prins [2014].

Both IRPs and PVRPs lie somewhere in the middle between tactical and operational levels. Most papers study these problems at an operational level. However, there are examples of real-world applications (Gaur and Fisher [2004]) that consider a medium to long time horizon. The integration of production and routing decisions is a topic of increasing interest. A survey of optimisation models dedicated to the study on this integration can be found in (Díaz-Madroñero et al. [2015]).

Clearly, tactical and operational decisions are not independent from each other. Tactical decisions strongly influence the operational level, conditioning the operational plans for a medium to long time horizon. On the other hand, tactical decisions should be taken considering operational models. For this reason, many solution methods for tactical problems make use of models and techniques used in operational problems (see, amongst other examples, Kilby and Urli [2016]; Smilowitz et al. [2013]; Schneider et al. [2014]; Gaur and Fisher [2004]).

Chapter 3

Solution Methods

In this chapter an overview of the methods used to solve VRPs is given, with a particular focus on the techniques that are used in Part II. The first two sections are dedicated to exact methods, and, in particular, to column generation. The last sections present an overview of heuristic methods, focusing in particular on local and meta-heuristic search.

3.1 Exact Methods

The research on exact methods for VRPs has been developed for over 40 years. The work done in this area is quite detailed and a survey is out of the scope of this section. Here, we briefly present the topic and refer to Semet et al. [2014] for a detailed overview on early methods, and to Poggi and Uchoa [2014] for an overview on new methods using set partitioning formulations.

The basic formulation for CVRP makes uses of binary flow variables x_{ij} to represent whether a vehicle travels on the arc connecting node i to node j for all $i, j \in \overline{C}\,(= C \cup \{0\})$. The problem can be formulated as follows:

$$\min \sum_{i,j\in\overline{C}} c_{ij}x_{ij} \tag{3.1}$$

$$\text{s.t.} \sum_{j\in\overline{C}} x_{ij} = 1 \qquad \forall\, i \in C \tag{3.2}$$

$$\sum_{j\in\overline{C}} x_{ji} = 1 \qquad \forall\, i \in C \tag{3.3}$$

$$\sum_{j\in\overline{C}} x_{0j} = |F| \tag{3.4}$$

$$\sum_{i\in S, j\notin S} x_{ij} \geq r(S) \qquad \forall S \subseteq C, S \neq \emptyset \tag{3.5}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i,j \in \overline{C} \tag{3.6}$$

The objective (3.1) minimises the total travelling costs. Note that, if $c_{ij} = c_{ji}$ for all $i, j \in C$, the CVRP is said to be symmetrical, otherwise it is said to be asymmetrical. Constraints (3.2) and (3.3) impose that each customer must be visited and must have exactly one predecessor and one successor. Constraint (3.4) sets the number of routes equals to the fleet size (note that a route can be empty). In constraints (3.5), $r(S)$ represents the minimum number of routes needed to satisfy the total demand of the subset S. In the CVRP, this can be computed through a bin packing problem. Usually, instead of $r(S)$, the lower bound $\lceil \sum_{i\in S} q_i / Q \rceil$ is used. Integrality of the solution is imposed by constraints (3.6). Note that constraints (3.5) serve at the same time as capacity and subtour elimination constraints. To see the former, consider S, the set of customers in a route r. The associated constraint (3.5) can be transformed into $\sum_{i\in S} q_i \leq Q$. A similar argument can be made for the subtour elimination constraints. This model was originally introduced in Laporte et al. [1986].

It is well known that the number of constraints (3.5) grows exponentially with N. Therefore, it is not practical, if at all possible, to include all of them in the formulation. One possibility is to solve the problem without such constraints, check if the solution violates some of them (a process named *separation*), in which case these are added to the formulation and the process is restarted. This procedure gives rise to cutting-plane algorithms. One other possibility is to include additional variables to represent the load (or the time, especially if one considers time windows) of a vehicle arriving at a customer's location. For example, if for each node $i \in \overline{C}$ we denote by u_i the load of the vehicle upon arrival at i, one would add the following constraints:

$$u_j - u_i \leq Q\,(1 - x_{ij}) - q_j \qquad \forall\, i,j \in \overline{C}$$

to replace the subtour elimination constraints, and

$$q_i \leq u_i \leq Q \qquad \forall\, i \in C$$

to replace the capacity constraints. The resulting model is known as *MTZ-formulation*. as it was originally proposed by Miller, Tucker and Zemlin in Miller et al. [1960]. This type of model is called *two-index* (vehicle) flow formulation. In some extensions of the CVRP, one may need to make the vehicles explicit in the flow variables, using therefore a *three-index* flow formulation.

This type of model is solved using branch-and-bound or branch-and-cut techniques, or mixtures of the two. A lot of effort has been spent on studying branching and lower bounding techniques to enhance the performance of the solution methods. Also, many different families of valid cuts have been introduced to strengthen the mathematical programming models for the CVRP, and many methods have been proposed for cut separation, i.e., to quickly identify violated cuts. An overview of this work can be found in Semet et al. [2014].

3.1.1 Set Partitioning Formulations

The Set Partitioning (SP), or Set Covering (SC), formulation for the CVRP was first introduced in Balinski and Quandt [1964]. Consider the set R of all possible routes for a CVRP instance. A binary variable x_r is associated to each route $r \in R$ to represent whether or not we use r. Moreover, the cost of route r is denoted by c_r, and, for each customer $i \in C$, the binary parameter a_{ri} denotes whether r visits i. The SP model follows:

$$\min \sum_{r \in R} c_r x_r \tag{3.7}$$

$$\text{s.t.} \sum_{r \in R} a_{ri}\, x_r = 1 \qquad \forall\, i \in C \tag{3.8}$$

$$\sum_{r \in R} x_r = |F| \tag{3.9}$$

$$x_r \in \{0, 1\} \qquad \forall\, r \in R \tag{3.10}$$

Constraints (3.8) impose that each customer is visited by exactly one route. Constraint (3.9) requires the number of selected routes to equal the fleet size. It is possible to replace the sign of this constraint to $\leq$ or to add $|F|$ copies of the empty route to R to allow for idle vehicles. Integrality of the variables is imposed by constraints (3.10). Since a solution of this model is, in particular, a partition of C, this formulation is named Set Partitioning formulation. By changing the sign of constraints 3.8 to $\geq$,

one obtains a Set Covering formulation. Note that from a solution of SC we can obtain a solution of SP by, possibly, dropping customers from routes. Moreover, if the triangular inequality for travelling costs is satisfied, dropping customers cannot result in an increase of the cost. Therefore, there exists an optimal solution of SC that is also an (optimal) solution for SP. The main reasons to use SC instead of SP are that, in the former, amongst routes with the same costs, only inclusion-maximal feasible circuits need to be considered. Moreover, as the dual variables associated with constraints (3.8) have to be non-negative in the SC dual problem, the dual space is considerably reduced.

The issue with SP or SC formulations lies in the huge number of variables. The enumeration of all possible routes is out of reach for reasonable sized problems. Consequently, it is not possible to solve the model directly. Instead, one has to resort to Column Generation (CG). Since CG is the topic of next section we postpone a few remarks on the use of CG in the routing literature. To conclude this section, we only mention that, since the seminal work of Desrosiers, Soumis and Desrochers (Desrosiers et al. [1984]), the SC formulation, in combination with CG, has been the dominant approach for exact methods for the VRPTW. According to Poggi and Uchoa [2014], this type of method can consistently solve instances of 200 customers. In Pecin et al. [2017], two instances of 200 customers each are solved with computational times of about 1, and 11 hours. Research to improve exact methods for CVRP and VRPTW is still ongoing (Uchoa et al. [2017]).

3.2 Column Generation

In this section, the main goal is to describe Column Generation and its relation to Dantzig-Wolfe decomposition. At the end of the section, we briefly overview the use of CG in the VRP literature.

Column Generation is a useful technique when a mathematical problem cannot be directly solved due to its high number of variables. Consider the following linear problem:

$$\textbf{[P]} \ \min c^\top x \tag{3.11}$$

$$\text{s.t. } Ax = a \tag{3.12}$$

$$x \geq 0 \tag{3.13}$$

where $x \in \mathbb{R}^n, a \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. At each iteration of the simplex method, only a few variables have non-zero value, i.e., the ones forming the basis. Moreover, many columns (variables) never even enter the basis. Therefore, the idea behind column

generation, is to avoid generating and considering these columns. We denote by A_j a column of problem **P**, and by c_j its cost coefficient. We define $J = \{1, \ldots, n\}$, the index set of all columns. The *master problem* is the following linear program, which is a reformulation of problem **P**:

$$(\mathbf{MP}) \min \sum_{j \in J} c_j x_j \tag{3.14}$$

$$\text{s.t.} \sum_{j \in J} A_j x_j = a \tag{3.15}$$

$$x_j \geq 0 \qquad \forall j \in J \tag{3.16}$$

In the simplex method, at each iteration, one looks for a non-basic variable $j \in J$ with negative reduced cost to enter the basis. If no such column is found, the current basic solution is optimal. CG mimics the simplex method, with the only difference being that, in CG, at each iteration, only a (small) subset $\overline{J}$ of J is available. The master problem, reformulated using only the subset $\overline{J}$ is called *restricted master problem*, and is denoted by **RMP** in what follows. We assume that the set $\overline{J}$ contains a feasible solution for **RMP**. If this cannot be guaranteed, then **RMP** has to be moified to ensure feasibility. At each CG iteration, **RMP** is solved, and the dual variables π, associated with constraints (3.15), are computed. Then, the goal is to identify a new column with negative reduced cost (note this cannot be in $\overline{J}$ by linear programming theory (Bertsimas and Tsitsiklis [1997])). In order to do this, one solves the following minimisation problem, called the *sub-problem*:

$$c^*(\pi) = \min\{c_j - \pi^\top A_j \mid j \in J\} \tag{3.17}$$

The objective in the sub-problem is precisely the current reduced cost of a column. Because the dual variables are sometimes called shadow prices, the sub-problem is also referred to as the *pricing-problem*. If the optimal value of the sub-problem is non-negative, the optimal solution of **RMP** is also optimal for **MP** and therefore for **P**. Otherwise, the minimising column is inserted in $\overline{J}$ and the process starts again. The CG algorithm has to be initialised with a starting set of columns. Usually, these are obtained using a heuristic method to "warm start" the solution. Algorithm 1 summarises the CG method.

We denote with z^* and $z^*_{\mathbf{RMP}}$ the optimal values of **MP** and **RMP**. At each iteration, the current value $z_{\mathbf{RMP}}$ of **RMP** is an upper bound for both quantities. Moreover, given that the cost of a linear problem cannot be reduced by more than $\sum_{j \in J} x_j$ times the minimum reduced cost $c^*(\pi)$ (Lübbecke and Desrosiers [2005]), a lower bound is

Algorithm 1 Column Generation

1: Fix a starting subset $\overline{J} \subseteq J$ of columns
2: Solve **RMP** using the columns in $\overline{J}$
3: Obtain the associated dual variables π
4: Solve the sub-problem using π
5: **if** $c^*(\pi) < 0$ **then**
6: Identify the column $\bar{j}$ that is the minimiser of the sub-problem
7: Add $\bar{j}$ to $\overline{J}$ and go back to Step 2
8: **else**
9: Stop. The current solution is optimal for **MP**.
10: **end if**

derived.

$$z_{\mathbf{RMP}} + c^*(\pi) \sum_{j \in J} x_j \leq z^*_{\mathbf{MP}} \leq z^*_{\mathbf{RMP}} \leq z_{\mathbf{RMP}} \tag{3.18}$$

The lower bound can be used to compute an optimality gap. Note that, if $c^*(\pi) = 0$, then $z^*_{\mathbf{MP}} = z_{\mathbf{RMP}}$ and optimality follows.

It is clear that a key part in the CG algorithm is Step 4. Indeed, one has to find an efficient way to generate new columns. Usually, the sub-problem has a special structure to make it possible to efficiently solve it without enumerating all columns. A relevant example, coming from the application of CG to VRPs, is provided later in Section 3.2.4.

3.2.1 Common Issues

There are many potential issues and delicate questions in the implementation of CG. In the following, we mention some of the issues that typically arise. We refer to Lübbecke and Desrosiers [2005] for a more detailed presentation.

Managing Columns. First of all, one has to decided how to manage the set $\overline{J}$. The number of generated columns can become very high and consequently slows down the solution of **MP**. Options range between keeping all generated columns and only the ones in the current optimal basis (Lübbecke and Desrosiers [2005]; Bertsimas and Tsitsiklis [1997]).

Master Problem One typical issue occurs in the solution of the master problem. A primal solution may have several correspondent dual solutions, especially if the problem is degenerate. This could lead to a poor choice of the dual solution, which in turn results in the generation of useless columns. Therefore, it is common to

solve the dual problem of **MP**. This also helps to overcome potential numerical issues in the computation of the dual variables and facilitates numerical stabilisation techniques, which are presented later in the section. Degeneracy is also a problem to be considered. A typical issue is that newly generated columns do not decrease the objective as the current solution is degenerate. This can slow down the process considerably.

Sub-Problem Another matter is the number of columns to be generated at each iteration and how they are generated. The generation of a column usually involves solving an integer problem. Several solution techniques, such as dynamic programming, might produce several columns with negative reduced cost. Therefore, one has to choose whether to insert all of them in $\bar{J}$, or choose only the one with smallest reduced cost. Moreover, the sub-problem does not have to be solved to optimality at each iteration, since we only need one column with negative reduced cost.

It is very common, to speed up the solution of the sub-problem, to use heuristic, especially in the first iterations. When these heuristics fail to obtain a column with negative reduced cost, the sub-problem is solved to optimality. Note that this is also needed to check the optimality of the current **MP** solution. The efficient solution of the sub-problem is arguably the most important step in CG. In Vanderbeck [1994], a concept of *dominance* for columns is introduced. This is meant to help excluding columns that cannot ever be optimal in the sub-problem because there will always be another column with lower reduced cost. All of these issues are discussed in more detail in Lübbecke and Desrosiers [2005].

Dual Variable Oscillations. In some applications, dual variables do not converge to their optimal value smoothly. Instead, they strongly oscillate in the process. This is regarded a major computational limitation (Lübbecke and Desrosiers [2005]). To overcome this problem, several stabilisation techniques have been proposed. These generally involve controlling the evolution of the dual solution, or, slightly modifying the value of the dual variables when solving the sub-problem. In Marsten [1975], the authors propose the "boxstep" method. The idea is to limit the variation of the dual variables in a small box around their current value, so that oscillations are prevented. In Madsen [1975], it is proposed to automatically adjust the size of the box based on how well the dual restricted master problem is approximating the Lagrangian dual problem. This method is called the proximal trust-region method. Other variations of this type of method have been studied (Amor and Desrosiers [2006]; Du Merle et al. [1999]).

Tailing-off Effect. Another important issue is the tailing-off effect. In the last iterations, the columns found have negative reduced costs close to zero, causing small or possibly null improvements in the objective. This is also due to the poor convergence properties of the simplex method. Other solution methods for the master-problem can help closing the optimality gap faster. The matter is discussed in more detail in Lübbecke and Desrosiers [2005]. Moreover, if CG is embedded in a branch-and-bound algorithm, lower bounds are used to prematurely stop the execution of a node. Typical lower bounds are lagrangian-type bounds that are based on equation (3.18).

3.2.2 Dantzig-Wolfe Decomposition

We now focus on the relation between CG and the Dantzig-Wolfe decomposition (Dantzig and Wolfe [1960]). Consider the following linear problem:

$$\begin{aligned} (\mathbf{O})\ \min\ & d^\top x \\ \text{s.t. } & Bx = b \\ & Dx = d \\ & x \geq 0 \end{aligned} \tag{3.19}$$

which we refer to as the *original formulation*. To keep the presentation concise, we assume that the solution space of problem **O** is bounded. This assumption is not necessary and easily removed (see Lübbecke and Desrosiers [2005]; Bertsimas and Tsitsiklis [1997]). Nonetheless, it is satisfied by the problems considered in Part II. We define the following set, $X = \{x \in \mathbb{R}^n \mid Dx = d, x \geq 0\} \neq \emptyset$ and let $\{x_j\}_{j \in E}$ be the set of extreme points of X. It is well known that any $x \in X$ can be represented as a convex combination of its extreme points. That is,

$$x = \sum_{j \in E} \lambda_j x_j \qquad \text{where} \qquad \sum_{j \in E} \lambda_j = 1 \text{ and } \lambda_j \geq 0,\ \forall j \in E$$

We set $c_j = c^\top x_j$ and $B_j = B\, x_j$. By using the above equation, problem **O** can be reformulated as

$$
\begin{aligned}
(\textbf{DW-O})\ \min & \sum_{j \in E} c_j \lambda_j & & \\
\text{s.t.} & \sum_{j \in E} B_j \lambda_j = b & & (3.20)\\
& \sum_{j \in E} \lambda_j = 1 & & (3.21)\\
& \lambda_j \geq 0 & \forall\, j \in E &
\end{aligned}
$$

This plays the role of the master problem in the previous section. Note that only the constraints associated with matrix B are part of the new formulation. The ones associated with D are now included in the definition of the variables. Since the size of X is usually huge, one resorts to using CG. The columns of this new formulation are the extreme points $\{x_j\}_{j \in E}$. Therefore, it is essential for the set X to have a special structure. We denote by α the vector of dual variables associated with constraints (3.20) and with θ the dual variable associated with constraint (3.21) (which is called the *convexity constraint*). The sub-problem can be written as:

$$\min\{c_j - \alpha^\top B_j - \theta \mid j \in E\}$$

Now that the master and sub-problem are defined, CG can be applied. Note that, thanks to the convexity constraint, the lower bound (3.18) now takes the form

$$z_{\textbf{RMP}} + c^*(\pi) \leq z^*_{\textbf{MP}} \leq z_{\textbf{RMP}}.$$

In the literature, the constraints defined by the matrix B are called the complicating constraints, and define the master problem. On the other side, the ones defined by the matrix D are used to define the structure of the sub-problem. Mathematically speaking, the Dantzig-Wolfe decomposition can be described as follows: given a linear problem, one splits the set of constraints in two parts, a set of complicating constraints and a set of constraints that are used to define a new search space. This latter set defines a polygon P in the original search space. Note that every point in the original search space is contained in P. Since every point in P can be expressed as the convex combination of its extreme points, the Dantzig-Wolfe decomposition formulates a new problem using as variables the extreme points of this polygon. This new formulation involves the complicating constraints, which are used to enforce feasibility of the solution, and a convexity constraint, used to enforce convex combinations on the extreme points. Since the number of extreme points is usually astronomical, CG is used to solve the newly formulated problem. For this whole

process to have some benefits, P must have a special structure that one can exploit to efficiently find extreme points.

In many applications, the matrix D has a diagonal structure

$$D = \begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^k \end{pmatrix}, \quad d = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{pmatrix}$$

where each block $D_i \in \mathbb{R}^{m \times n_i}$ and $\sum_{i=1}^{k} n_i = n$. If one defines

$$X^i = \{x \in \mathbb{R}^{n^i} \mid D^i x = d^i, x \geq 0\}, \quad i = 1, \ldots, k$$

the Dantzig-Wolfe decomposition can still be used, except that now we have k sets. Geometrically, polygon P is decomposed[1] into k polygons, each one associated with matrix D_i, and with its own extreme points. Therefore, we can split the sub-problem into k separated sub-problems. The CG algorithm does not change. At each iteration, we solve k sub-problems to identify new columns. Note that not all of the sub-problems have to be solved at each iteration, as only one column with negative reduced cost is needed. However, all of them has to be solved to optimality to prove the master problem optimality (Step 9). Finally, if $c_i^*(\pi)$ denotes the value of the i-th sub-problem, the lower bound (3.18) can be rewritten as:

$$z_{\mathbf{RMP}} + \sum_{i=1}^{k} c_i^*(\pi) \leq z_{\mathbf{MP}}^* \leq z_{\mathbf{RMP}}.$$

3.2.3 Integer Problems

Because one needs the dual variables to solve the sub-problem, the master problem (and thus the original problem **P**) has to be linear. If the original problem **O** is integer, one has to add the integrality constraint

$$\sum_{j \in E} \lambda_j x_j \in \mathbb{Z}^n$$

to the problem **DW-O**. In this case, CG has to be embedded in a branch-and-bound scheme, obtaining what is called a *branch-and-price* (BAP) method. If valid cuts are progressively added to the formulation, sometimes the method is called *branch-and-cut-and-price*. One interesting remark is on the branching strategy. This must not dis-

[1] The polygon P is being projected on the sub-spaces of dimensions n_i associated with matrixes D_i.

rupt the structure in the sub-problem, causing any difficulty in efficiently solving the sub-problem. An example, clarifying this concept, is presented later in Section 3.2.4.

3.2.4 Column Generation Applications in Routing Problems

When applied to VRPs, CG, in combination with the SP (or SC) formulation has proved very successful. The idea was first applied to VRPTW in Desrosiers et al. [1984]. The SP formulation (3.7-3.10) can be obtained by applying the Dantzig-Wolfe decomposition to the flow formulation (3.1-3.6), by moving the subtour elimination constraints (3.5) into the sub-problems, and by dividing constraints (3.3) and (3.2)) between master and sub-problem. In the master problem, one keeps the requirement to visit all customers, while in the sub-problem one imposes flow conservation constraints. The polygon P, defining the sub-problem search space, is made up by all feasible routes. The sub-problem takes the form of a (Elementary) Shortest Path Problem with Resourced Constraints (SPPRC, Irnich and Desaulniers [2005]). SPPRCs are usually solved by means of a forward dynamic programming labelling algorithm (Ahuja et al. [1993]). In a nutshell, the idea is to create partial paths starting from the depot and progressively extend them to other nodes. Paths are eliminated based on dominance rules. A dominance rule's job is to make sure one does not keep memory of sub-optimal paths. More details can be found in Irnich and Desaulniers [2005]; Ahuja et al. [1993].

As mentioned before, since VRPs are integer problems, some care must be taken when defining a branching strategy for BAP methods. For example, fixing a variable $x_r = 0$ in the master problem destroys the sub-problem structure, or, to be more precise, our ability to efficiently solve it. Therefore, typically, the branching is performed on arcs, even if these are not variables in the master problem. This type of branching rule can be included more effectively in dynamic programming methods for SPPRCs.

The research on CG for VRPs has focused on possible ways to improve BAP algorithms: valid cuts, sophisticated branching rules and efficient labelling algorithms have been proposed and studied. One notable issue is the possibility of routes with cycles when solving the sub-problem. A route is an elementary path, i.e. it does not contain cycles. However, in the sub-problem, some arcs might have a negative weight, due to high values of some dual variables. This can lead to the generation of routes with cycles, since dynamic programming methods do not naturally prevent this from happening. Although given a path with cycles, one can easily obtain a route by dropping customers that are visited more than once, this results in lower performance. One possibility is obviously to modify the labelling algorithm so as to eliminate partial paths that contain cycles. However, this considerably slows down

the algorithm. Some authors eliminate only 2-cycles (Christofides et al. [1981]), which has been shown to improve performance. Other authors have introduced a relaxed concept of "route". For example, in Christofides et al. [1981]; Fukasawa et al. [2006], the authors introduce the concept of q-route; that is, a route that is allowed to visit a customer twice, but only if at least another k customers are visited between the repeated visits. In a post processing phase, q-routes can be easily transformed in elementary routes. For a more detailed presentation we refer to Poggi and Uchoa [2014].

In the literature concerning routing problems, several heuristic methods are based on CG (Mourgaya and Vanderbeck [2007]; Ceselli et al. [2009]; Bertoli et al. [2017b,c]; Taillard [1999]). A classification of CG-based heuristics can be found in Joncour et al. [2010]. In Rousseau et al. [2004], the authors propose using Constraint Programming (Rossi et al. [2006]) to solve the pricing problems.

Finally, it is worth mentioning that CG-based algorithms have successfully been applied into a variety of routing problems such as: fleet size and mix problems (Choi and Tcha [2007]; Baldacci and Mingozzi [2009]), SDVRPs (Archetti et al. [2015a]), dynamic routing (Chen and Xu [2006]), PVRPS (Mourgaya and Vanderbeck [2007]), IRPs (Le et al. [2013]; Oppen et al. [2010]) and rich VRPs (Ceselli et al. [2009]). Many papers do not attempt at solving a problem exactly but only propose heuristics based on the CG algorithm.

3.3 Heuristic Methods

Despite the research effort dedicated to exact methods, these are able to solve consistently only relatively small instances of simple VRPs, and need a long time to do so. Real-world problems are often very large and characterised by several constraints that make the problem harder to capture and solve with mathematical programming tools. For these reasons, much effort has been dedicated to the development of heuristic methods. The literature on this topic is vast, and many different methods have been proposed and studied. In this section, our goal is to give a brief presentation of some heuristic and metaheuristic methods. We go into more detail only for methods that are used in Part II. We refer to Gendreau and Potvin [2010] for a general review of metaheuristics. For surveys focusing on heuristic methods for VRPs, we refer to Laporte and Semet [2002]; Gendreau et al. [2002]; Laporte et al. [2014]. A survey on metaheuristics for the VRPTW can be found in Bräysy and Gendreau [2005b]. In the next sections, we present some of the main references and results for constructive heuristics, improvement heuristics and metaheuristics for the class of VRPs. We focus in particular on the methods that we use in in Part II.

3.3.1 Constructive Heuristic

Constructive heuristics are mostly used to provide a starting solution for improvement heuristics. One of first methods proposed was the Clarke and Wright's Saving heuristic (Clarke and Wright [1964]). The idea is very simple: one first builds a route of the form $(0, i, 0)$ for each customer $i \in C$. Successively, the algorithm merges pairs of routes. To be precise, merging a route $(0, \ldots, i, 0)$ to $(0, j, \ldots, 0)$ results in the route $(0, \ldots, i, j, \ldots, 0)$, i.e., routes are connected end to start. The pair to be merged is chosen, among all pairs that would yield a feasible route, based on the savings obtained by merging the two routes.

Other constructive method are based on the *route-first cluster-second* algorithm, originally proposed in Beasley [1983]. The idea is to first solve a TSP for all customers and then split the so obtained giant-tour in smaller routes. However, this type of method is seldom used. A contrasting principle is the *cluster-first route-second* principle. Algorithms of this type attempt to first create clusters of customers and then solve a TSP for each cluster. Two well-known examples are the Fisher and Jaikumar method (Fisher and Jaikumar [1981]) and the sweep heuristic (Gillett and Miller [1974]). In the former, a partition of the customers is obtained by solving an assignment problem with an objective function approximating the routing cost. In the latter, clusters are formed based on the polar coordinates of the customers' location.

Insertion heuristics constitute another type of constructive methods. The typical insertion heuristic is structured as follows: first it chooses a seed customer, usually the one farthest from the depot, or the most urgent one (if time windows are considered), to initialise a route. Successively, customers are chosen and inserted in the route, one at the time. If no more feasible insertion is found, the algorithm "opens" a new route by choosing a new, still unrouted, seed customer and the insertion process starts again. Candidate insertions are evaluated with metrics combining geographical and other information, e.g., time-related. One of the most successful insertion heuristics for the VRPTW is the I1 method, originally proposed by Solomon (Solomon [1987]). The method evaluates the best insertion of each customer using a metric accounting for distance saving and time delay due to the insertion. Successively, the insertion to be performed is chosen using a different metric, that rewards customers far from the depot. For a more detailed presentation of the I1 heuristic and related modifications, we refer to Bräysy and Gendreau [2005a]. These types of insertion heuristic are called *sequential*, as they build one route at the time. Conversely, there are *parallel* insertion heuristics, where m routes are initialised with seed customers. The routes are then built in parallel, i.e., one can choose the route to insert a new customer.

3.3.2 Improvement Heuristics

Improvement heuristics attempts to improve a given solution by modifying its structure. Based on an initial solution to start the improvement process. Classical methods perform intra-route and inter-routes moves. Most of the intra-route moves were originally designed for the TSP. Examples are the 2-opt and λ-opt moves (Bräysy and Gendreau [2005a]; Toth and Vigo [2014]). Examples of inter-routes moves are the *relocate* move, where a customer is removed from its original position and relocated somewhere else; the *swap* move, where two customers are swapped; or the $2 - opt^*$ move, that is an extension of the 2-opt move involving two different routes (Toth and Vigo [2014]). Since all of these moves are formally defined, it is possible to exhaustively enumerate all the solutions that can be obtained applying one particular move. Such exploration can be computationally expensive. However, a smart implementation of the exploration can considerably reduce the number of possibilities to be evaluated. For example, in Savelsbergh [1992], the authors illustrate how to speed-up the search of the neighbourhood defined by a 2-opt for a TSP solution.

Other types of method are not defined by specific moves but attempt to improve a solution by alternating methods that partially destroy a solution (i.e., they remove customers from it), and then repair the solution. A well-known example is the Large Neighbourhood Search heuristic (Shaw [1998]), which we present in detail later in this chapter.

3.4 Metaheuristic

Broadly speaking, metaheuristics can be described as high-level procedures that are used to explore the solution space to find sufficiently good solutions. They often use standard constructive and improvement heuristics to do so. Metaheuristics are radically different from classical early approaches in that they allow for deteriorating and sometimes infeasible solutions in order to better explore the search space and to escape from local minima. Metaheuristics used to solve VRPs can be broadly split in two classes: local search-based and population-based methods. The main difference between the two classes is in the exploration of the search space: local search-based methods move from one solution to another one, while population-based methods evolve a set (population) of solutions, by combining or modifying them, with the goal of progressively generating better solutions. Examples of local search-based metaheuristics are: iterated local search (Lourenço et al. [2010]), variable neighbourhood search (Hansen and Mladenović [2014]), simulated annealing (Nikolaev and Jacobson [2010]), tabu search (Glover [1986]), large neighbourhood search (Shaw [1998]),

guided local search (Voudouris and Tsang [2003]), hill climbing and late acceptance hill climbing (Burke and Bykov [2012]). The class of population-based metaheuristics includes: genetic algorithms (Holland [1992]; Lin [1965]; Prins [2004]), scatter search and path relinking (Resende et al. [2010]), swarm intelligence and ant colony optimisation (Dorigo et al. [2006]). This division is not rigid and some metaheuristic try to draw concepts from both classes. One example is the Adaptive Memory Programming (Taillard et al. [2001]) approach, which combines Tabu Search (TS) and genetic algorithms. The combination of population-based algorithms with methods that exploit specific knowledge of the problem gives rise to memetic algorithms, see Moscato and Cotta [2010] for more details.

The rest of this section is dedicated to the Large Neighbourhood Search (LNS) and Adaptive LNS (ALNS) metaheuristics. These are the two metaheuristics used in the heuristic algorithms presented in Part II.

3.4.1 Local Search and Large Neighbourhood Search

As already stated, local search-based algorithms explore the solution space by moving from one solution to another. They start from an initial solution s_1, which can be generated randomly or by a constructive heuristic. At each iteration t, the algorithm move from the current solution, s_t, to another solution, s_{t+1}, that belongs to its neighbourhood $N(s_t)$. Note that it can happen that $s_{t+1} = s_t$. Moreover, deteriorating solutions can be allowed, i.e., if we denote by $f(s)$ the cost of a solution s, it could be that $f(s_{t+1}) > f(s_t)$. The neighbourhoods of a solution and how to move to a new solution depend on the particular algorithm chosen. Since metaheuristics do not know whether they have reached the optimal solution or not (unless some lower bounds are available), there is no mathematically sound stopping criterion. Hence, most metaheuristics impose a maximum number of iterations or a maximum execution time.

We now focus on the Large Neighbourhood Search metaheuristic, originally proposed by Shaw (Shaw [1998]). Let us introduce some notation to formally present the algorithm. We write S to denote the solution space and $f(s)$ to denote the cost of a solution $s \in S$. We assume S is finite. Our goal is to find the optimal solution, i.e., $s^* \in S$ such that $f(s^*) \leq f(s)\ \forall s \in S$. A neighbourhood of $s \in S$ is defined as a subset $N(s) \subseteq S$. A local optimum is defined as a solution $\bar{s}$ such that $f(\bar{s}) \leq f(s) \forall s \in N(\bar{s})$. LNS is a "destroy and repair" metaheuristic; that is, one defines a destroy and a repair heuristic, denoted by d and r. Given a solution s, the destroy heuristic removes some customers from s. Conversely, the repair heuristic re-inserts the removed customers into s. Typically, the destroy method contains some

randomness, so that different customers can be removed every time the method is invoked. The neighbourhood $N(s)$ is then defined as the set of solutions that can be obtained by first applying the destroy method and then the repair method. Since the destroy method could remove several customers, the size of $N(s)$ can be quite large, which explain the name of the metaheuristic. As an example, a very simple destroy method randomly selects a fixed percentage of customers to remove while a repair method could insert the customers using a greedy heuristic. The pseudo-code for LNS is given in Algorithm 2.

Algorithm 2 LNS

1: Create an initial solution s
2: $s_{\text{best}} = s$
3: **repeat**
4: $s' = r(d(s))$
5: **if** AcceptanceCondition(s', s) **then**
6: $s = s'$
7: **end if**
8: **if** $f(s') < f(s_{\text{best}})$ **then**
9: $s_{\text{best}} = s'$
10: **end if**
11: **until** stopping condition is met
12: **return** s_{best}

While exploring the space, LNS keeps memory of the best solution encountered, which is returned as the output of the algorithm. The implementation of the AcceptanceCondition in Step 5 is left open. It is up to the user to decide whether to accept only improving moves, or to allow for deteriorating solutions. If S is the space of all feasible solutions, the feasibility of the current solution s has to be maintained by the operator r. However, in principle, S can contain infeasible solutions. If these are highly penalised by the objective function the output of LNS will be feasible (provided the initial solution is feasible). The reason for allowing infeasibility is to make the algorithm able to escape local minima. However, one of the major advantage of LNS lies in the fact that by considering large neighbourhoods it is less likely to be trapped in local minima. Moreover, as already pointed out, another mechanism to escape from local minima is to accept deteriorating solutions. Therefore, many implementations only consider feasible solutions. The typical stopping criteria are a pre-fixed maximum number of iterations, or a time-out. A more detailed presentation and several applications of LNS can be found in Pisinger and Ropke [2010].

We now focus on ALNS. This was originally proposed in Ropke and Pisinger [2006] and constitutes an extension of LNS. The main difference lies in the fact that ALNS uses adaptive weights to choose amongst several destroy heuristics $d_1, \ldots, d_{n_d}$

and several repair heuristics $r_1, \ldots, r_{n_r}$. A positive weight is assigned to each destroy/repair method. At each iteration, the destroy and repair methods to be used are randomly selected according to their weights. A higher weight means a higher probability of being chosen. One key feature of ALNS is that the weights are progressively adjusted based on how well the associated heuristics have performed in the past. Let us denote by w_{d_i} the weight associated with destroy heuristic d_i and similarly for the repair heuristics. The pseudo-code for ALNS is provided in Algorithm 3.

Algorithm 3 Basic ALNS

1: Create an initial solution s
2: $s_{\text{best}} = s$
3: $w_{d_i} = 1, \forall i = 1, \ldots, n_d$, $w_{r_i} = 1, \forall i = 1, \ldots, n_r$
4: $it = 0$
5: **repeat**
6: **for** M iterations **do**
7: $it = it + 1$
8: Sample a destroy heuristic d_i and a repair heuristic r_j
9: $s' = r_j(d_i(s))$
10: **if** AcceptanceCondition(s', s, it) **then**
11: $s = s'$
12: **end if**
13: **if** $f(s') < f(s_{\text{best}})$ **then**
14: $s_{\text{best}} = s'$
15: **end if**
16: **end for**
17: Update weights of destroy/repair heuristics
18: **until** stopping condition is met

The standard acceptance criterion used for ALNS uses the SA metaheuristic. This means all improving solutions are accepted, whereas deteriorating solutions are accepted with probability $e^{-\Delta/T_t}$ where Δ is the cost gap $\Delta = f(s_{t+1}) - f(s_t)$ and T_t is the current temperature. The temperature is initialised to a value T_0 and the decreased gradually according to a cooling schedule. One common example is to set $T_{t+1} = \alpha T_t$ where $\alpha \in [0, 1]$. Usually the initial temperature is set to a value comparable with the cost of the initial solution. The idea is that, as the search progresses, the algorithm accepts deteriorating solutions less and less frequently, converging to a hill climbing acceptance method.

In Step 8, the destroy/repair methods are selected following a *roulette wheel principle*. This can be easily described as follows: the probability $p(d_i)$ of choosing

method d_i is computed normalising its weight, i.e.,

$$p(d_i) = \frac{w_{d_i}}{\sum_{i=1}^{n_d} w_{d_i}}$$

The same can be done for the repair methods. The adaptive adjustment is done every M iterations (what is called a *segment*), corresponding to the inner loop in Algorithm 3. We illustrate how the weights are adjusted only for the destroy heuristics, but the same applies for the repair heuristics. Let us define a cumulative score $\Psi(d_i)$ for each destroy heuristic d_i. At the start of a segment, we set $\Psi(d_i) = 0$ for all $i = 1, \ldots, n_d$. After a new solution s' is generated using d_i, in Step 9, $\Psi(d_i)$ is increased by

- ψ_1, if s' is the new global best
- ψ_2, if s' has not been accepted before and it improves the current solution s
- ψ_3, if s' has not been accepted before and it does not improve the current solution s but is accepted

The adjustment of the weights has two goals: reward heuristics that improved the current and best solutions and reward heuristics that help diversify the search. It is well known that diversifying the search is a key concept in local search-based algorithms. Finally, in Step 17, the weights are computed as follows:

$$w_{d_i} = (1 - \lambda)\, w_{d_i} + \lambda \frac{\Psi(d_i)}{t_{d_i}} \quad \forall i = 1, n_d$$

where t_{d_i} is the number of times the heuristic d_i has been called in the last segment. The *decay* parameter $\lambda \in [0, 1]$ controls the sensitivity of the weights to the performance of the associated heuristics. On one end, if $\lambda = 0$, the weights are constants. On the other end, if $\lambda = 1$, the weights are completely defined by results in the last segment.

The implementation of the destroy and repair heuristic depends on the particular problem being solved. Some of the most common destroy methods follow. In all of them, the number of customers to be removed is randomly chosen at every invocation.

- Random removal: it randomly selects and removes some customers.
- Worst removal: it removes the customers whose removal most decreases the objective function.

- Related removal: it removes a set of customers that are related to each other. It first selects a random customer and then ranks the remaining ones using a relatedness metric.

- Historical node-pair removal: it exploits historical information to decide what customers to remove. A score is associated with each edge and is updated throughout the algorithm depending on how often the edge appears in a good solution. The scores are used to select the customers to be removed.

All of the above operators can be modified to make use of the specific knowledge of the problem, for example using time-related information in the case of the VRPTW. Moreover, some randomness can be introduced in the selection of customers. For example, in the worst removal method, a parameter can be introduced to control the probability of always choosing the worst customer, or less expensive customers. These and other destroy heuristics are described in detail in Pisinger and Ropke [2007]. Repair heuristics are typically greedy or k-regret insertion heuristics. We refer to Ropke and Pisinger [2006]; Pisinger and Ropke [2010]; Shaw [1998]; Pisinger and Ropke [2007] for a more detailed presentation of ALNS.

Finally, we mention that in their paper (Ropke and Pisinger [2006]), the authors added some noise to the objective function to further randomise the search. However, this is not a recurring feature in ALNS implementations that have been proposed in the literature.

3.5 Matheuristic

Some methods proposed in the literature hybridize metaheuristics to mathematical programming methods. These types of methods are sometimes called *matheuristics* (see Maniezzo et al. [2010]). We have already mentioned the existence of CG-based heuristics for routing problems in Section 3.2.4. A classification of such heuristics is proposed in Joncour et al. [2010]. In this section, we briefly touch upon the subject of matheuristics without aiming for a detailed presentation. In Boschetti et al. [2009], the authors define matheuristics as follows:

> *matheuristics are heuristic algorithms made by the interoperation of metaheuristics and mathematical programming techniques.*

Two general surveys can be found in Puchinger and Raidl [2005]; Ball [2011]. In Archetti and Speranza [2014], a survey of matheuristics for routing problems is

presented. In Doerner and Schmid [2010], the authors review the use of matheuristics for rich VRPs.

In Part II, some of the proposed methods may be classified as matheuristics. In this section, our only goal is to summarise the type of methods that are used. In each chapter of Part II, we give relevant references. The methods proposed in Chapter 4 are based on CG. Following the nomenclature suggested in Joncour et al. [2010], they are a restricted master heuristic and a branch-&-price heuristic. In Chapter 5, the numerical analysis is done using exact methods. However, in order to speed-up the process, we propose a simple matheuristic to warm-start the proposed MIP. Such method is based on the SC model for VRPs, presented in Section 3.1.1. The idea is the following: we run ALNS on the problem at hand and record every best solution found. At the end of the algorithm we formulate and solve a SC-type model using the routes that make up the solutions recorded by ALNS. The main idea behind this algorithm is to use a heuristic method to explore the space and identify promising regions. Successively, a MIP is used to intensify the search in the explored region. According to Archetti and Speranza [2014], all methods we develop can be classified as set-covering/partitioning-based approaches. Many methods proposed in the literature (Archetti et al. [2008]; Subramanian et al. [2013]; Renaud et al. [1996a]) use this type of approach for several variants of the VRP. A more detailed presentation of the methods is given in each chapters of Part II.

Part II

Tactical Routing Problems

Chapter 4

Fleet Design

4.1 Introduction

Road freight transport is a fundamental part of every country's economy. According to Rodrigue et al. [2006], it is not unusual for transportation costs to account for 20% of the final cost of a product. In Australia, the transportation sector employs 1.6% of all workers and, between 2014 and 2015, generated an annual revenue of $52 billion (Bankwest [2015]). However, the profit margins have not changed since 2006 (Bankwest [2015]). The growth in economy and consumption increases the need for efficient transportation. The competition faced by transportation providers increases the pressure for reduced costs.

Parallel to the increase in the demand and productivity is the growth in the number of freight vehicles. Companies face fleet selection and dimensioning problems at all decision levels. A major goal both for transportation providers and goods owners is to strike the optimal balance between costs due to the acquisition and maintenance of a fleet, costs of daily operations and costs due to the subcontracting of the deliveries. Market-related information such as transportation rates, vehicle costs, and future expected demand play a key role in fleet sizing decisions.

Fleet design is the problem of determining the size and composition of a fleet of vehicles to carry out the daily delivery operations of a company. The problem can be informally stated as follows: given

- the daily demand of a set of customers over a reasonably long period of time, i.e., the horizon,
- a "catalogue" of truck types with different characteristics, e.g., capacity, running costs, compartments, etc., and
- a model describing the constraints and costs incurred in running the daily delivery operations,

identify the most efficient fleet to satisfy the demand across the whole horizon.

Most freight forwarding companies face strong fluctuations in the market which result in a varying demand over time. On the one hand, they have to ensure that enough resources will be available to cover a day's orders. On the other hand, the fixed costs of vehicles (e.g., insurance and amortisation costs) force them to keep their fleet small in order to reach a high utilisation of the available vehicles. Usually, only a part of the upcoming requests is fulfilled by owned transportation resources. All the remaining orders are outsourced or, alternatively, processed using hired vehicles. When designing a fleet for a long time-horizon, companies should consider these options, as they allow them to keep the fixed costs low without diminishing the capability of providing an efficient delivery service.

Even though most companies build their fleet slowly over a long time-horizon, there are situations where a company needs to design an entire fleet, or a sub-part of it, from scratch. Examples include fleet down-sizing after a merger, or fleet acquisition in response to signing a new, large contract. A company may also want to determine the ideal fleet, so as to have a "target fleet" to guide future purchases.

In the operations research literature, fleet design is mainly studied either as a strategic or as an operational problem. In the former case, the horizon is very long and the data is characterised by high uncertainty. In such cases it is difficult to account for future routing costs, hence these are usually neglected, or approximated according to delivery area size (Jabali et al. [2012]), and the focus is on the fleet decisions only. If routing information is not available, a common approach is to estimate, e.g. through a probability distribution, the demand over a certain horizon. In these cases, the goal is to design a fleet able to satisfy, possibly with a certain probability, the demand. On the other side, a large part of the literature considers fleet design from an operational point of view, that is, as a simple extension of the classic VRP, where the fleet is not given as input but is a decision to be made. In this case the time horizon is rarely longer than a day, due to the complexity of the problem. We will refer to this class of problems as Fleet Size and Mix (FSM, Golden et al. [1984]; Toth and Vigo [2014]; Baldacci et al. [2008]). In a tactical setting, the situation is different. More reliable data is usually available and it is desirable to consider routing costs as these will represent a significant part of the cost of operating the fleet. To the best of our knowledge, there is no paper attempting to study the tactical version of fleet design on a multi-day horizon considering both the possibility of acquisition and hiring of vehicles.

The rest of the chapter is structured as follows: in Section 4.2 we illustrate the real-world applications that led us to focus on fleet design problems. In Section 4.3, we give a formal description of the problem studied. Successively, in Section 4.4,

the relevant literature is reviewed. Section 4.5 is devoted to the description of the models and the solution methods proposed. In Section 4.6 we give a detailed account of the algorithms implementation. Experimental results and analysis are presented in Section 4.7. We finally conclude summarising the contributions contained in this chapter and pointing out some future research directions.

The material in this chapter has been presented in Bertoli et al. [2017b], where the tactical fleet design problem was introduced, and in Bertoli et al. [2017a], where the possibility of hiring vehicles was included.

4.2 Motivation

In a previous collaboration with an industrial partner, our research group faced a real-world fleet design problem. The problem and the solution method proposed are described in Kilby and Urli [2016]. In this paper, the authors formulate the problem as multi-day FSM and propose an approach based on a pre-processing step that identifies a set of Pareto dominant days. The idea is that, if a fleet can cover a "big-demand" day, then it will also be able to cover all "smaller" days. This is possible in the specific problem presented because a dominance rule, i.e., a partial order between days, can be established. While the approach proved to be viable for the problem at hand, three major issues can be identified: scalability, efficiency and, more importantly, generality. The scalability issue arises from the fact that the whole multi-day problem associated with the Pareto front has to be solved at once in order to guarantee that, on each day, a subset of the same fleet of vehicles is used. Given that the single-day problem is already **NP**-hard, solving a multi-day variant for a large Pareto front can easily become intractable.

An additional inherent problem is the efficiency of the fleet. Since the fleet produced will be tailored for big days only, the authors have shown there will often be a high number of idle vehicles per day, which is not desirable.

A more fundamental problem is generality, that is, the problem-specific pre-processing technique cannot be extended to other routing problems. For example, this approach does not work when compatibility constraints are considered. The fleet for the Pareto days could, and in fact often will, be infeasible for days with smaller demand but with more compatibility constraints. The situation is even more complex when time related constraints, such as time windows, are considered. Defining a dominance rule for rich VRPs is not trivial, if at all possible. Moreover, the method cannot take into account the possibility of hiring extra vehicles or subcontracting.

For all these reasons, we wished to develop a different, more general approach to fleet design. In particular, we were presented with a new real-world fleet design

problem, which is described in Section 4.7. Even if the goal is still to design an efficient fleet, the underlying routing problem, i.e. the set of operational constraints, is different from the problem considered in Kilby and Urli [2016]. In fact, the technique proposed in Kilby and Urli [2016] cannot be applied to such a problem. An attempt to extend it was done, but with poor results. In Section 4.5.6, we formally describe the technique presented in Kilby and Urli [2016], why it cannot be easily generalised and elaborate on how the new method we proposed can be seen as an evolution of such a technique.

The methods we developed to solve this new problem are very general and can be applied, under some assumptions, regardless of the particular underlying routing problem. Therefore, this chapter presents such methods from a general point of view. We assume, however, that we are able to have a suitable solver, able to solve the FSM problem for a single day. This is clarified when presenting the solution methods. In Section 4.7, we describe the real-world application motivating this study.

4.3 Problem Formulation

The problem considered in this chapter is that of designing a fleet that minimises the sum of operational costs and fixed costs over a multi-day[1] planning horizon while satisfying the demand of all customers, and meeting all the operational constraints on each day. We allow purchase, sale, and rental of vehicles. The only subcontracting option we consider is to hire extra vehicles. Hiring an extra vehicle can result in a fixed price and/or a variable price that depends on the usage of the vehicles[2]. Therefore, the fixed costs are the sum of purchase costs, fixed prices of vehicles' rental, minus the profits coming from the sale of vehicles. The operational costs are the costs related to routing the vehicles. These may be dependent on the vehicle type and on whether the vehicle is owned or hired.

Formally, let D be the set of days, or *horizon*, that we are considering, and T be the set of all available vehicle *types*. Each vehicle type $t \in T$ has a fixed cost b_t, which may aggregate, for instance, acquisition and maintenance costs. Once acquired, a vehicle can be used every day: we pay b_t if we use the vehicle once, regardless of how many days the vehicle is used subsequently. We denote by E_t the number of vehicles of type $t \in T$ that we already own. We will refer to both the vehicles that we buy and that we already own as the *owned* vehicles. It is also possible to sell some of these vehicles. The revenue from selling a vehicle of type t is s_t. For consistency,

[1] As stated in Section 2.3 the term "day" should be understood to indicate a period, i.e.,"a convenient unit of time".

[2] We will see that some other subcontracting options can be easily included in our model. However, for sake of clarity we postpone this to later sections.

we assume $s_t < b_t$ as otherwise it would be convenient to buy and then sell vehicles. We assume that all of the fleet changes are made at the beginning of the horizon. Therefore, a vehicle which is sold cannot be used on any day. On the other hand, a vehicle that is bought is considered available on every day.

We also assume that the horizon is divided into P shorter consecutive intervals (of days) P_p, $p = 1, \ldots, P$, that represent the minimum hiring time period, e.g. a week. With a slight abuse of notation we also denote by P the set $\{1, \ldots, P\}$. The collection $\{P_p\}_{p \in P}$ is a partition of D where each set contains consecutive days. For example, if D represents a month, the intervals could be weeks or single days. On a given interval P_p we can hire a vehicle of type $t \in T$ by paying a fixed price h_{tp}. We assume that $h_{tp} \geq b_t \frac{|P_p|}{|D|}$ so that the rental is in general less convenient than purchase if we intend to use a vehicle for the whole horizon. This is a reasonable assumption, as hiring vehicles is used as a way to accommodate extra demand on a given day, rather than the primary option for transportation. If this is not the case, then the problem can be reformulated by looking at the different intervals separately, and considering only the hired vehicles. Note that, for each vehicle type $t \in T$, the operation cost of using a hired or owned vehicle for the same route could be different. This happens, for example, if a hired vehicle has a higher cost per unit of distance or if a vehicle is hired at a flat daily rate.

On each day $d \in D$ we have to satisfy the demand of a given set of customers C_d. To keep the notation simple, we differentiate between requests of the same customer over different days, and we consider them as different customers. In detail, given two different days $d_1, d_2 \in D$, there might be customers $i \in C_{d_1}$ and $j \in C_{d_2}$ such that i and j share the same location. However, they are considered as different customers. This way, a customer is naturally associated with a unique day. We denote the set of all customers on the horizon by $C = \cup_d C_d$.

We also are given a (possibly rich) set of constraints, e.g., time windows, pickup and delivery constraints, compartments constraints, vehicle-commodity compatibility constraints, etc. One key aspect of our approach is that it does not depend on the specific day-to-day constraints considered. The only limitation is that we do not consider inter-route constraints such as synchronisation of routes[3]. We denote by VRP(d) the (rich) vehicle routing problem for day $d \in D$.

We also assume the existence of a solver for the FSM extension of VRP(d), that is able to choose an efficient fleet, and efficient routes, to serve the customers in C_d according to the operational constraints that apply.

The goal is to find a fleet that minimises the sum of the operational and fixed costs. Note that by *fixed costs* we mean both the costs and revenues due to acquisition

[3]The reason for this requirement is given in Section 4.5

(b_t), sale (s_t), and hiring (h_{tp}) of vehicles.

The problem described is very realistic as it considers sale, purchase, rental of vehicles, the existence of a prior fleet, but also, and especially, because the routing constraints are not specified.

4.4 Related Work

Now that we have stated the problem, we review the relevant literature. In Section 2.3, we already discussed how transportation and fleet design problems are classified based on the three decision levels: strategic, tactical and operational. Detailed reviews on the literature of all three levels can be found in Hoff et al. [2010] with particular focus on industrial applications. Since our work can be placed at the tactical level, we focus mostly on this level.

As pointed out in Hoff et al. [2010] and Salhi and Rand [1993], there is not much work focusing on tactical fleet design problems. We quote Hoff et al. [2010]:

> *A large part of the literature focuses on operational questions, along the line of "what to do given a certain fleet mix and a given set of service request". This is in contrast to the more tactical, or strategic, "which vehicles should we acquire to best solve our daily routing problem for the next half year". There is a big absence of papers addressing these more tactical questions, and also on how to make robust or resilient solutions.*

We refer to Hoff et al. [2010]; Baldacci et al. [2008] for a survey on fleet design from respectively, an industrial and an academic point of view. We only focus on few exceptions that can be considered belonging to the tactical level.

In Salhi and Rand [1993], the authors note the same lack in the literature and develop an advanced heuristic. Their solution method is based on a route perturbation procedure. The overall performance is good and the algorithm proved to be stable and able to handle complex constraints but is not tested on problems with a long horizon. In Yoshizaki et al. [2009], a real-world fleet design problem is considered. The horizon is a week-long. The authors solve the problem by applying a scatter search-based heuristic. However, they do not solve the whole horizon at once but each day separately. The daily solutions can then be used to inform decisions for the whole horizon.

As we noted before, a first paper attempting to solve on a year worth of data is Kilby and Urli [2016]. We already presented the main idea of this paper and we postpone a deeper discussion to Section 4.5.6.

Most of the existing literature considers fleet design as an extension of the classic VRP, where the fleet composition is not an input to the problem, but rather a decision to be made. There are several variants of this extension depending on whether the fleet is unlimited or not and the cost are vehicle dependent or not (see Baldacci et al. [2008]; Toth and Vigo [2014] for surveys on this class of problems).

Due to its huge search space, the FSM is much harder to solve than its VRP counterpart. For this reason, the FSMs models rarely extend beyond considering a single representative day of demand data, or model not-so-rich VRP variants. Perhaps the maximum demand, or the 80^{th} percentile is chosen. However, this approach cannot adequately represent the variability in demand day to day and season to season. In addition, rich constraints may make a good solution for one day infeasible for another. For example, a good solution for a "big day" scenario may use many large trucks. But vehicle compatibility constraints, which forbid large vehicles visiting some customers, may make that solution expensive, or even infeasible, for a day with more moderate demand.

Another method, typical of stochastic optimisation (M. Gendreau [2014]), is to analyse the demand patterns of each customer, and fit a probability distribution function. We can then use this to determine the probability of needing to visit a customer, and the probability of exceeding capacity. However, in both cases, this in effect imposes a "grand tour" constraint: customers, if they require service, are visited in the same order every day. Moreover, if time windows are present, the grand tour must observe the time windows in each scenario (or in expectation), further constraining the routes. If such a "grand tour" constraint is not present in our problem, enforcing it can lead us to very sub-optimal solutions. Another stochastic optimisation approach is to choose a set of "representative" scenarios, and evaluate the objective on these scenarios. This approach is the closest analogy for the method described here, although we use *all* days as scenarios, and then use factors emerging from the mathematical analysis to reduce the scenarios actually evaluated. Two important advantages of our approach over these two approaches concern efficiency and feasibility. On one side, the risk in choosing the representative days is to underrepresent easy and small days, this leads to big fleets, efficient on big days but having a high number of idle vehicles on small days. On the other side, choosing a subset of representative scenarios can become problematic when attributes other than demand are considered. As we point out in this work, if compatibility constraints, time windows or other operational constraints are included in the model, it is harder to choose a set of good representative scenarios that guarantee the fleet will be enough (feasible) for all scenarios. This may lead to an overuse of hired vehicles, which is sub-optimal.

Finally, we focus on the literature considering the problem of balancing owned vehicles against the subcontracting of customers to external transportation providers. This problem arises in different areas of the literature. In Loxton et al. [2012], the authors consider the problem of designing a fleet over a multi-period horizon, considering also the option of hiring trucks. The problem is seen as a stochastic one, where the number of vehicles needed on a period is a random variable. The goal is to determine a fleet composition so that the expected sum of fixed costs and hiring cost is minimised. In their model, hiring trucks can be seen as a recourse action. This type of approach can be seen as belonging to the strategic level. We highlight that the authors do not consider operational costs. In Chu [2005], the authors introduce the so called Integrated Operational Transportation Problem (IOTP). The goal in the IOTP is to route a fixed fleet and determine which customers will be served by the fleet and which will be subcontracted to external carriers for a fixed price per customer. The authors solve the problem in a hierarchical fashion by firstly splitting the customers into two sets. The first set is composed of the customers that will be served by the owned fleet, for which they solve a standard routing problem. The second set is composed of the customers to be served by hired trucks, for which the cost is easily computed. A step forward is taken in Kopfer and Wang [2009] where the IOTP is extended to the Vehicle Routing with Forwarding Problem (VRFP). The main difference lies in the fact that there is more than one type of subcontracting option. The authors develop a MIP and solve rather small instances of the problems, although these are sufficient to show the potential savings due to mixing different subcontraction options. Moreover, they point out the importance of tactical decisions when designing a fleet, and the importance of considering the option of subcontracting customers. A survey on the IOTP and VRFP can be found in Kopfer and Krajewska [2007].

The IOTP and VRFP can be seen as a generalisation of the FSM problem. In the latter, there is no distinction between hired and owned vehicles, and one needs only to determine the number and type of vehicles to include in the fleet. In the former problems, not only does one have to decide which type of vehicles, but also which customer to subcontract and, in the VRFP, also what type of subcontracting option to choose. However, the problems considered in the literature are mostly limited to one day of operations and are of rather small size. In Kopfer and Wang [2009] the authors consider the problem on a multi-day horizon. However, the instances are very small and they are solved considering each day separately, therefore not tackling the problem as a whole.

4.5 Solution Methods

In this section, we present three solution methods for the described problem. Before presenting the methods in detail, we try to give a brief overview of the solution framework. The problem we are considering naturally decomposes over the days of the horizon. What tie the different days together is the constraint to have same fleet and the associated fixed cost. The main idea is to decompose the problems over the days using a CG framework (see Section 3.2). The sub-problems' role is to generate new *operational solutions* for each day. On the other hand, the master problem chooses which daily solution to use, and the best overall fleet compatible with such daily plans. In the following, we describe in details what we mean by daily solutions and provide mathematical models for the master and the sub-problems.

4.5.1 Fleet Generation

Given two vectors $q, w \in \mathbb{R}^T$, we define $\text{FSM}(d)(q, w)$ to be the FSM problem where

- the operational constraints and the operational costs of the vehicles are the ones given by $\text{VRP}(d)$;
- for each vehicle type t we have two corresponding vehicles types: one for the owned vehicle and one for the hired vehicle;
- the fixed prices are given by the two vectors $q, w \in \mathbb{R}^T$ for, respectively, owned and hired vehicles;
- we have unlimited availability of vehicles.

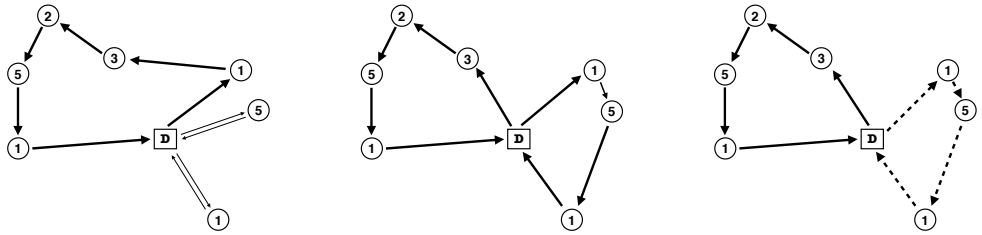
Briefly, $\text{FSM}(d)(\cdot, \cdot)$ is a FSM problem where we are duplicating each vehicle type and differentiating on its status: owned or hired. We assume we have available a solver for the $\text{FSM}(d)(\cdot, \cdot)$ problem.

Assumption 1. We have available a solver for the FSM extension of $\text{VRP}(\cdot)$. We denote such solver as FSM-solver.

For now, we leave the choice of such solver open. In Section 4.7, when presenting the real-world application we use to run our tests, we also describe the FSM-solver that we use.

For each day $d \in D$, we denote by X_d the set of all solutions of $\text{FSM}(d)(\cdot, \cdot)$. Note that X_d does not depend on the fixed costs. We refer to the elements of X_d as *routing plans*. A routing plan s for day $d \in D$ is identified by its operational cost (i.e., the operational costs deriving from routing both the owned and hired vehicles), c_{ds}, and

by the number of owned and hired vehicles of type t that we denote by, respectively, F^t_{ds} and H^t_{ds}. We give an illustrative example of this concept in Figure 4.1.



(a) Solution 1 (b) Solution 2 (c) Solution 3

Figure 4.1: Suppose we have only two types of vehicle, with capacity 20 and 5, respectively represented by the thick and thin lines. Continuous and dotted lines are for owned and hired vehicles respectively. Solutions 1 and 2 have only owned vehicles while solution 3 has one hired vehicle.

Let us introduce some notation. For each $t \in T$, the integer variables F^+_t and F^-_t represent, respectively, the number of vehicles of type t we buy and sell. Moreover, for each $p \in P$, the integer variable H_{tp} represents the number of vehicles of type t we hire in interval p.

For each day $d \in D$ and each $s \in X_d$ we introduce a binary decision variable x_{ds} representing whether that routing plan is actuated or not. We model the problem as follows:

$$\textbf{(F)}\ \text{minimise} \sum_t F^+_t\, b_t - \sum_t F^-_t\, s_t + \sum_{t,p} H_{tp}\, h_{tp} + \sum_{d,s} c_{ds}\, x_{ds} \tag{4.1}$$

$$\text{subject to} \sum_s x_{ds} = 1 \qquad \forall\, d \in D \tag{4.2}$$

$$\sum_s F^t_{ds}\, x_{ds} \leq F^+_t - F^-_t + E_t \qquad \forall\, d \in D, t \in T \tag{4.3}$$

$$\sum_s H^t_{ds}\, x_{ds} \leq H_{tp} \qquad \forall\, p \in P, d \in P_p, t \in T \tag{4.4}$$

$$F^-_t \leq E_t \qquad \forall\, t \in T \tag{4.5}$$

$$F^+_t, F^-_t, H_{tp} \geq 0,\ \textit{integer} \qquad \forall\, t \in T, p \in P \tag{4.6}$$

$$x_{ds} \in \{0,1\} \qquad \forall\, d \in D, s \in X_d \tag{4.7}$$

The objective function (4.1) has two components modelling fixed and operations costs. The fixed costs component is in turn divided into the cost for buying, revenue from selling, and cost of hiring, vehicles.

Constraints (4.2) mean that we can only select one routing plan for each day.

Constraints (4.3) ensure that the overall fleet, as a result of acquisition and sale of vehicles, has enough vehicles of type t to actuate the chosen routing plan each day. Similarly, constraints (4.4) ensure that we hire enough vehicles over a given interval, so that each day of that interval we can actuate the chosen routing plan. Constraints (4.5) mean we cannot sell more vehicles than we actually have in the fleet. Constraints (4.6) and (4.7) enforce the integrality of the solution.

Model **F** is a Set Partitioning formulation for the LHFSM. Note that, since we are solving a minimisation problem, we can replace constraints (4.2) by $\sum_{s \in X_d} x_{ds} \geq 1$, making F a Set Covering formulation. However, note that in this particular case overcovering cannot happen, as choosing more than one routing plan per day is sub-optimal. Moreover, we can replace constraints (4.7) with $x_{ds} \geq 0$, *integer*. Finally, note that if we consider single days as intervals for hiring, constraints (4.4) can be removed as $H_{tp} = \sum_{s \in X_d} H^t_{ds} x_{ds}, \forall p, t$. In Bertoli et al. [2017b], we studied the same problems only considering the purchase of vehicles. The model presented in Bertoli et al. [2017b] can obtained from **F** by setting $E_t = F^-_t = H_{tp} = 0$ for all $t \in T, p \in P$.

Note that even the simple enumeration of all routing plans for a given day is computationally challenging. Since we also need to know the operational cost c_{ds} of a routing plan, it is clear we cannot list all possibilities for all days. This issue is solved with the use of column generation. Let us denote by **LF** the linear relaxation of **F** and by π_d, θ_{td} and η_{td} the dual variables of constraints (4.2), (4.3) and (4.4). For an interval $p \in P$ and a day $d \in P_p$, the reduced cost of a variable x_{ds} is

$$c_{ds} + \sum_{t \in T} F^t_{ds} \, \theta_{td} + \sum_{t \in T} H^t_{ds} \, \eta_{td} - \pi_d. \tag{4.8}$$

The goal of the sub-problem is to generate a column whose reduced cost is negative. The variables in the sub-problems are the cost c_{ds} and the values F^t_{ds}, H^t_{ds}, which uniquely identify a column. The dual variables of the master are considered as input parameters in the sub-problem.

If we exclude term π_d, which is constant, the sub-problem becomes exactly problem FSM$(d)([\theta_{td} \mid t \in T], [\eta_{td} \mid t \in T])$, i.e., a FSM problem where the fixed costs are given by the dual variables of the master problem. Note that, if operating an owned or hired vehicle of a certain type has the same cost, then we can reduce the dimension of the sub-problem by considering only one type based on whichever between θ_{td} and η_{td} is smaller. However, if hired and owned vehicles have different operational costs, then they have to be considered as two completely distinct types in the sub-problem. Note that model **F** is the result of a Dantzig-Wolfe decomposition applied to a standard flow formulation of the problem if we take the solutions spaces of FSM$(d)(\cdot, \cdot)$ as the sets X^i in Section 3.2.

The first method we propose is a simple CG-based heuristic. We initialise **LF** with a column for each day by solving FSM$(d)(0,0)$. In other words, the first column for a day d corresponds to the sub-problem where $\theta_{td} = \eta_{td} = 0 \,\forall\, t$. We then run column generation until a termination criterion is met (in the tests we will use a time limit). Once the execution of CG is over, we solve the integer version **F** with the columns found so far. Following the nomenclature proposed in Joncour et al. [2010], the CG-based heuristic we propose is a *restricted master heuristic*. We name this method Fleet Generation (FG).

One weakness of the formulation **F**, is the fact that we are duplicating the types of vehicles in the sub-problem. The FSM problem is known to be $\mathcal{NP}$-hard and adding vehicle types can only bring more symmetry to an already hard problem. We try to address this in the more sophisticated methods following.

In Bertoli et al. [2017b] we proposed, in addition to the same restricted master heuristic, a branch-&-price approach. However, the results indicated that this requires significantly more computational time and the gain in quality over FG is minimal. Therefore, in order to keep this presentation concise, we do not consider the branch-&-price algorithm here.

4.5.2 Route Generation

An alternative to generating entire routing plans, as in the model **F**, is to first generate *routes*, and then assign vehicles types (either hired or owned) to each route. This reflects the fact that, to some extent, a route can be efficient regardless of whether an owned or a hired vehicle is used. Indeed, the strength of the approach described here is that it avoids exploring the search space of routes for hired vehicles, and then exploring exactly the same space of routes with owned vehicles. This can be seen as avoiding a kind of symmetry within the route space.

The idea is to generate potential routes as a sub-problem, and then assign vehicle types in the master problem. Dual variables from the master problem will give the "value" of customers, leading to a sub-problem that can be seen as an elementary shortest path problem. As we mentioned in Section 3.2.4, applications of CG to classical VRPs generally follow this type of approach.

In order to keep our presentation as general as possible, we make only one requirement for the underlying daily problem: that it has no inter-route constraints. In CG applications to routing problems, a variety of intra-route constraints, such as time windows, and compatibility constraints can be easily handled while solving the sub-problem. However, inter-route constraints, such as *synchronisation* constraints (Drexl [2012b]), must be modelled at the master problem level. We therefore impose

this restriction in order to have a consistent master model. Later (in Section 4.7), we will see that some inter-route constraints, such as split deliveries or delivery of multiple commodities can be readily accommodated.

Again, for ease of presentation, let us assume, for now, that we only have one commodity and vehicles can perform one trip per day. We introduce the following notation.

- R represents the set of all possible routes on the horizon, a route r is associated to only one type $t \in T$ and one day $d \in D$;
- R_d represents the set of routes on day $d \in D$;
- R_{td} represents the set of routes of type $t \in T$ on day $d \in D$;
- a_{ir} is a binary parameter representing whether customer i is visited by route r;
- x_r^b and x_r^h are binary variables representing whether route r is performed with an owned or a hired vehicle, respectively;
- c_r^b and c_r^h are the costs of route r when this is executed with an owned or hired vehicle, respectively;

The model follows.

$$\text{(R) minimise } \sum_t F_t^+ b_t - \sum_t F_t^- s_t + \sum_{t,p} H_{tp} h_{tp} + \sum_r (c_r^b x_r^b + c_r^h x_r^h) \tag{4.9}$$

$$\text{subject to } \sum_r a_{ir}(x_r^b + x_r^h) = 1 \qquad \forall i \in C \tag{4.10}$$

$$E_t + F_t^+ - F_t^- \geq \sum_{r \in R_{td}} x_r^b \qquad \forall\, d \in D, t \in T \tag{4.11}$$

$$H_{tp} \geq \sum_{r \in R_{td}} x_r^h \qquad \forall\, t \in T, p \in P, d \in P_p \tag{4.12}$$

$$x_r^b, x_r^h \in \{0,1\} \qquad \forall\, r \in R \tag{4.13}$$

$$F_t^+, F_t^-, H_{tp} \geq 0, integer \qquad \forall\, d \in D, t \in T, p \in P \tag{4.14}$$

The objective (4.9) is the sum of the fixed costs and the operational costs; the only difference with model **F** is in how the operational cost is computed. Constraints (4.10) require that each customer is visited by one vehicle. Constraints (4.11) are analogous to constraints (4.3) for model **F**. They ensure the overall fleet is big enough to execute all the chosen route of a certain type in a specific day. Similarly constraints (4.12) make sure we hire enough vehicles on each interval. Finally constraints (4.13)

and (4.14) enforce integrality of variables. Similary to model **F**, the sign of constraints (4.10) can be changed to $\geq$, making model **R** a covering model. It is well known that from a solution of a covering model we can easily obtain a solution of a partitioning model by dropping requests.

Analogously with model **F**, it is not practical to enumerate all the variables (columns) of this model and, therefore, we rely on CG to solve it. We call **LR** the linear version of **R** and $\hat{\pi}_i, \hat{\theta}_{td}$ and $\hat{\eta}_{td}$ the dual variables of, respectively, constraints (4.10), (4.11) and (4.12). Given a route $r \in R_{td}$, the reduced cost of variables x_r^b and x_r^h are, respectively

$$c_r^b - \sum_{i \in C_d} a_{ir} \hat{\pi}_i + \hat{\theta}_{td} \qquad \text{and} \qquad c_r^h - \sum_{i \in C_d} a_{ir} \hat{\pi}_i + \hat{\eta}_{td} \tag{4.15}$$

If we ignore $\hat{\theta}_{td}$ and $\hat{\eta}_{td}$, which can be considered constant, the sub-problems become elementary shortest path problems with resources constraints. As we discussed in Section 3.2, typically, these problems are solved by means of labelling algorithms (Ahuja et al. [1993]). However, labelling algorithms do not suit every variation of these problems, especially the ones derived as sub-problems of rich VRP's formulations. As an example, when dealing with multiple compartments and multiple commodities, CG approaches have not been very successful: in Archetti et al. [2015a]; Mirzaei and Wøhlk [2016], the use of CG is limited to specific scenarios and tested only on quite small instances. The effectiveness of the method mainly depends on the strength of the partial path dominance rule. If this is able to eliminate many partial paths, then a labelling algorithm is likely to be very efficient. However, if the dominance rule is weak, the algorithm lists almost all the possible paths. Thus, CG cannot be effectively applied, for now, to all the variants of the VRP.

One other possibility is to view the sub-problem as a generalisation of the profitable tour problem (PTP, Tsiligirides [1984]; Archetti et al. [2014b]). In this problem, one has to design a route (tour) which respects all the operational constraints but does not have to visit all the customers. However, visiting a customer is rewarded with a (customer dependent) revenue. The goal is to minimise the operational cost of the route minus the total revenue. In this interpretation, the dual variables $\hat{\pi}_i$ can be seen as the profits. It can be easier to modify the available FSM-solver to account for profits and to relax the constraint that enforces service to all customers than to design an effective labelling algorithm. If this can be done, then, by considering only one vehicle of a given type t on a fixed day d, we can solve the associated sub-problems.

Similarly, to what we did previously for the FSM-solver, we assume the following

Assumption 2. We assume we have available a solver able to solve the sub-problems

(eq 4.15). We denote such solver as R-solver.

In Section 4.7, we opt for the second choice and modify our FSM-solver since our problem presents many complicating constraints that cannot be easily accommodated with a labelling algorithm.

The restricted master problem is initialised as follows: we solve the FSM$(d)(0,\infty)$ problem for each single day $d \in D$, i.e., we do not consider hired vehicles. This is done using the FSM-solver. Then we take as starting columns all the routes composing the best solution found on each day. We then run CG until optimality or a termination criteria is reached (in the simulation it will be a time limit). Once the CG execution is stopped we solve **R** with the columns obtained so far. We denote this method as "route generation" (RG). This is another example of a restricted master heuristic.

Note that there are still two different sub-problems for hired and owned types if the operational costs are different. Unfortunately, we are not able to eliminate a priori one of the two. This is due to the fact that different operational costs lead to different c_r^b and c_r^h for the same route r and that $\hat{\theta}_{td}$ and $\hat{\eta}_{td}$ are, in general, different. However, this does not pose a particular difficulty. Indeed, we can solve one sub-problem first, and we move on to the second only if the first has not produced new variables[4]. Moreover, note that a route is not associated a priori with an owned or hired vehicle but this is a decision variable in the model. Therefore, for example, once generated for a sub-problem related to owned vehicles, the route can still be executed by a hired vehicle.

While this approach breaks the symmetry present in the FG methods, one potential problem is that it could be slow. Route generation is a successful method in routing problems and there exist applications on problems with multi-day horizon (Le et al. [2013]). However, the horizon usually considered is small (a few days or a week) and the method has never been applied to a rich problem with a very long horizon and multiple vehicles types. We will study this potential issue in the experiments.

4.5.3 Refinement method

The third method combines FG and RG in a two-phase method. The intuition is that the FG process produces a large number of routes quickly, and that these can act as an effective pool to initialise the RG process. Furthermore, our observation that a route is effective regardless of whether it is performed by an owned or hired vehicle

[4]Which problem to solve first is discussed later in Section 4.6

means that we can ignore hiring options for the FG process. Ignoring hiring also eliminates the "route symmetry" problem discussed in Section 4.5.2.

We therefore first solve the original problem, without consider hiring as an option, using the FG method. This can be done by setting the variables H_{tp} to zero in model **F**. Every distinct route generated during this process is saved. Once we reach a termination criterion we move to the second phase. Here, we take all the routes generated in the first phase and use them to initialise model **R**. We can then proceed to apply the RG method. In other words, we use FG on a smaller problem to initialise RG. We denote this method as "refinement method" (RM).

4.5.4 Model Extension For Rich Routing Constraints

In Section 4.7, we apply the described methods to a routing problem that presents complicating constraints such as split deliveries, multiple commodities and multiple trips for each route. In this sub-section, we show how these can be easily incorporated in model **R**. Note that model **F** does not need any modification as the constraints are part of the underlying sub-problems, whose definition we left open. To support split deliveries, we simply allow parameters a_{ir} in model **R** to be continuous rather than binary. The modification needed to accommodate multiple commodities and multiple trips per vehicle are straightforward as well. Suppose we have U different commodities to deliver. Instead on having only one coefficient a_{ir} we consider U coefficients $a_{iru}, u = 1, \ldots, U$, one for each commodity. Analogously, we decouple constraints (4.10) in Y different constraints, one for each commodity

$$\sum_{r \in R} a_{iru}(x_r^b + x_r^h) = 1 \quad \forall i \in C, u = 1, \ldots, U.$$

Furthermore, suppose that each vehicle can perform Y possible trips, where each trip has to happen within a given time window, morning and afternoon in our case. Note that by a "trip" we define a route happening in a defined time window. It is not important whether the vehicle can return to the depot to refill, as this is seen as part of the route. We define R_{tdy}, instead of R_{td}, as the set of routes of vehicles of type t, on day d, performing trip y. We only need to decouple constraints (4.11) and (4.12) over the different trips. For example, constraints (4.11) become

$$E_t + F_t^+ - F_t^- \geq \sum_{r \in R_{tdy}} x_r^b \quad \forall\, d \in D, t \in T, y \in Y$$

Note that the sub-problems in method RG are not affected by any of these modifications.

4.5.5 Including other subcontracting modalities

We now want to briefly present how it would be possible to introduce new type of subcontracting option into our problem. In Section 4.3, we assumed that we can only hire extra vehicles. In Chu [2005], the authors consider what they call less-than-truckload (LTL) requests. This means that we can "trade" a customer to an external carrier, i.e. we pay a fixed price to an external carrier that is now responsible to deliver the customer's demand. Each customer may have a different price based on distance or other factors.

This can be easily introduced in model **R** by introducing new variables y_i for each customer $i \in C$ representing whether a customer is served with a hired or owned vehicle or by an external carrier. The only changes to make are in the objective function, that now has to account for this new source of cost, and in constraints (4.10), whose right hand sides would now be y_i.

If it is possible to modify the FSM-solver to account for profits, it is also possible to include LTL orders in the model **F**. Suppose that the LTL fee for a customer i is denoted as f_i. We can add $\sum_{i \in C_i} f_i$ to the objective in the sub-problem and set the profit of customer $i \in C_d$ equals to $-f_i$. The costs due to LTL deliveries would be now included in the operational cost of a routing plan and would not be included in the master problem.

Another subcontracting option introduced in Kopfer and Wang [2009] is to pay a fixed price for a route which will be executed by an external carrier. The route may be subject to some distance limitation. This type of constraints falls in the definition of VRP(d), which we left open. If the underlying solver supports them, then it is immediate to include this subcontracting option in the problem by setting the operational cost of a hired vehicle to zero and considering only the fixed prices h_{tp}.

4.5.6 An Extension of the Pareto Approach

In this section, we illustrate why the pre-processing technique presented in Kilby and Urli [2016] cannot be extended to other problems with more complicated constraints. We also elaborate on how the mathematical programming framework that we are using allow us to gain some insights on how to extend the idea of identifying a subset of critical days.

For sake of clarity, let us assume the underlying routing problem, VRP$(\cdot)$, has the following features: 2 commodities to be delivered, 2 types of truck, one dedicated to each commodity, and capacity constraints only. Let us denote by q_i^1, q_i^2 the demand of the two commodities of customer $i \in C$. A day d can be associated with the

pair $(q_d^1, q_d^2) = (\sum_{i \in C_{d_1}} q_i^1, \sum_{i \in C_{d_2}} q_i^2)$. The pre-processing technique is based on the following dominance rule: consider two days $d_1, d_2 \in D$, if $q_{d_1}^1 \geq q_{d_2}^1$ and $q_{d_1}^2 \geq q_{d_2}^2$, then *any* fleet that can be used on day d_1 can also be used on day d_2. This dominance rule defines a partial order. If we identify the Pareto front, i.e., the set of non-dominated days, we can solve the fleet design problem *only* using the days in the front. Note that, to solve the problem on the Pareto front we need to solve a multi-day FSM. The so obtained fleet is guaranteed to be feasible on every day. This is true only because we are only considering capacity constraints. If we consider time windows or driver-customer compatibility constraints in VRP(·), there is no easy way to define such a dominance rule. One could hope that these constraints do not play an important role and leave the dominance rule unchanged. Unfortunately, we tried this approach but it led to a significant number of unfeasible days.

In order to have a clearer presentation let us reformulate model **F** without considering any prior fleet and without considering hired vehicles, i.e., we are only allowed to buy vehicles. The model then becomes

$$
\begin{aligned}
\text{minimise } & \sum_t F_t^+ \, b_t + \sum_{d,s} c_{ds} \, x_{ds} & & \\
\text{subject to } & \sum_s x_{ds} = 1 & \forall\, d \in D & \\
& \sum_s F_{ds}^t \, x_{ds} \leq F_t^+ & \forall\, d \in D, t \in T & \qquad (4.16) \\
& F_t^+ \geq 0, \mathit{integer} & \forall\, t \in T & \\
& x_{ds} \in \{0, 1\} & \forall\, d \in D, s \in X_d &
\end{aligned}
$$

By using the same notation for the dual variables, we can write the dual model of the linear relaxation of the model above

$$
\begin{aligned}
\text{maximise } & \sum_d \pi_d & & \\
\text{subject to } & \sum_d \theta_{td} \leq b_t & \forall\, t \in T & \\
& \pi_d - \sum_t F_{ds}^t \, \theta_{td} \leq c_{ds} & \forall\, d \in D, s \in X_d & \\
& \pi_d \in \mathbb{R} & \forall\, d \in D & \\
& \theta_{td} \geq 0 & \forall\, d \in D, t \in T &
\end{aligned}
$$

We recall that the dual variables θ_{td} can be interpreted as daily fixed costs. The dual problem "distributes" the costs b_t over the days with the goal of maximising

the sum of the lower bounds (π_d in this context) on each day's total cost. If, for a fixed $d \in D$ and $t \in T$, Constraint (4.16) is not tight, the Complementary Slackness Theorem implies that $\theta_{td} = 0$, and, therefore, the vehicle is "free" in the sub-problem. Intuitively, the variables θ_{td} tell us how much a vehicle of type t is worth on day d if we can only use the columns generated so far. In fact, at each iteration of CG, there will be only a few days having some of the θ_{td} that are non-zero. Note that, due to how we initialise the model, sub-problems with all free vehicles do not need to be solved again. This information points us to the days that need other routing plans (columns) in order to reduce the overall fixed costs and it is of fundamental importance for an efficient implementation as is discussed in next section. In some sense, the dual variables identify a subset of critical days. Notably, this set changes at each iteration, as the insertion of new variables in the master changes the dual problem. However, the main advantage of our decomposition is that we can solve each day independently. Oppositely, in Kilby and Urli [2016] one has to solve a multi-day FSM, considerably increasing the complexity of the method.

4.6 Implementation Details

In this section, we focus on a few issues and details related to the implementation of the algorithms.

4.6.1 Theoretical caveats.

Let us only consider FG and model **F**. The fact that the sub-problem is solved heuristically creates a theoretical issue that, however, can be easily addressed in the implementation. The heuristic might find the same column (routing plan) multiple times with different routing costs. This is because the heuristic might find a better way to route the same fleet. If the cost is lower than the previously found one, we just replace the existing column with the newly found one. Otherwise we discard it.

4.6.2 Termination Criterion

In cases where optimality is too hard to obtain, it is common to prematurely stop the CG algorithm when a termination criterion is met. A common approach is to compute a lower bound, usually based on a Lagrangian relaxation, and stop the execution whenever the gap between the linear objective and the lower bound is smaller than a pre-specified threshold. In our case it is not easy to compute, in a reasonable time, a reliable lower bound as this requires a good lower bound on the

FSM problem on each day. For these reason, and to allow a fair comparison between the various methods we chose to run each algorithm for a limited amount of time.

4.6.3 Sub-problem Selection

One important aspect to consider is the number of problems to be solved at each iteration. Note that in the column generation phase, we do not need to solve all the sub-problems, but only enough to generate a new column. The number of sub-problem is higher for model **R** than for model **F**. However, in the latter, the sub-problems are much harder. The number of the sub-problems to be solved is an issue to consider, especially given that, according to our experiments, 99% of the algorithmic time is spent solving the sub-problems. In the experiments, we solve only a limited number of sub-problems, denoted by ρ, at each iteration. Therefore, we try to solve ρ sub-problems. If we find a column with negative reduced cost, we insert it in the master problem and move on to the next iteration. If no such column is found, we solve another ρ sub-problems. After carrying out preliminary parameter tuning, we decided to set the parameter to $\rho = 10$ for the algorithm FG, and $\rho = 20|T|$ for RG.

A related issue is the selection of the sub-problems to be solved. A standard approach would be to compute a lower bound for the reduced costs. The variables with lowest lower bounds would be the ones that are more likely to have a negative reduced cost. Unfortunately, we do not have an efficient method to compute a reliable lower bound for any of the models **LF** and **LR**. Instead, the selection methods we use, which is similar for both models, exploit the information contained in the dual variables. At each iteration, we rank the sub-problems based on the current and past value of the dual variables, and then we chose the first ρ sub-problems in the rank.

Let us first illustrate the method for the FG algorithm. To keep the notation simple, let us assume that we can only buy vehicles, i.e. model **F** can be rewritten as done in Section 4.5.6. However, including the hired vehicles in the formula is straightforward. The aim is to diversify the search by avoiding solving a sub-problem similar to ones we have solved before. The idea is to compare the current value of dual variables θ_{td} with the values used in the past iterations where we solved the sub-problem for day d, and choose the sub-problem with the greatest difference. To achieve this, we rank the days using a weighted total variation. We fix a day $d \in D$ and denote by $\widetilde{X}_d$ all the routing plans $s \in X_d$ that we have generated so far. For each $s \in \widetilde{X}_d$, we denote by θ^s_{td} the value of the dual variables at the iteration where we generated the routing plan s, and by $|F_{ds}|$ the total number of vehicles in routing

plan s. The score of a day is then given by the following formula

$$\sum_{s \in \widetilde{X}_d} \sum_{t \in T} (\theta_{td} - \theta^s_{td})^2 \frac{F^t_{ds}}{|F_{ds}|} \frac{1}{|\widetilde{X}_d|}$$

Namely, we take the total variation with respect to the variables θ^s_{td}, weighted by the importance that vehicle type t had in the routing plan s. Alternatively, one could define the score as the minimum variation (i.e., replacing the sum over all routing plans with a minimum) or simply consider the sub-problems whose dual prices are the highest. However, experiments showed that taking the total variation produced better results.

For the RG algorithm, the ranking rule is slightly more involved. In this case we estimate the reduced costs and rank the sub-problems in ascending order. Let us introduce some notation. Each sub-problem can be associated with a triplet (d, t, v) with $d \in D, t \in T$ and v describing if the vehicle is owned (b) or hired (h). For a given day d and a given type t, let $\Gamma(d, t)$ be the average number of customers per route, taken over the routes in R_{td} that we have generated so far. Similarly, we denote by $\Theta^b(d, t)$ and $\Theta^h(d, t)$ the average of the cost of routes executed with owned and hired vehicles, respectively. We define $\Phi(d) = \sum_{i \in C_d} \pi_i / |C_d|$, i.e., the average of the current value of the dual variables for the customers on day d. We estimates the reduced cost of sub-problems (d, t, b) and (d, t, h) (eq (4.15)) as follows :

$$\gamma^b(d, t) = \Theta^b(d, t) - \Phi(d) * \Gamma(d, t) + \theta_{td}$$
$$\gamma^h(d, t) = \Theta^h(d, t) - \Phi(d) * \Gamma(d, t) + \eta_{td}$$

We are estimating each term in the formulas (4.15) by, respectively: the average of previous routes' cost, the average of previous routes' number of customers and the average of current dual values π. One possibility would be to select the ρ sub-problems (d, t, v) with lowest $\gamma^v(d, t)$ and solve these. However, we observed that this leads to the selection of the same types of vehicles in all iterations. Therefore, for each day we average over the vehicle types and obtain, for each of the pairs (d, b) and (d, h), the *scores*

$$\gamma^b(d) = \frac{\sum_{t \in T} \gamma^b(d, t)}{T}$$
$$\gamma^h(d) = \frac{\sum_{t \in T} \gamma^h(d, t)}{T}$$

Note these quantities are associated with pairs (d, v) and do not depend on vehicles type. We select the $\tilde{\rho}$ pairs with lowest scores. Each pair (d, v) is associated with $|T|$ sub-problems (one for each vehicle type). After preliminary parameter tuning, we set $\tilde{\rho} = 20$. Therefore, the number of sub-problems solved at each iteration is $\rho = 20|T|$.

4.7 Computational Analysis

In this section, we aim at analysing the performance on the proposed methods. We first present the real-world problem that motivated this work. This is the particular instance of $VRP(\cdot)$ we use in all our experiments. The data was provided by a business partner. The context is a fuel distribution problem in northern Australia. We briefly describe the underlying (rich) VRP, the FSM-solver and R-solver we use to solve the sub-problems in FG and RG.

Routing problem and solver The underlying routing problem can be summarised as follow: every day we have a set of requests to be satisfied by a fleet starting at and returning to a single depot. Each request is characterised by *i)* the demand, which involves two different commodities, *ii)* a time window (morning, afternoon, or the whole day), *iii)* a service time which depends on the quantity being delivered, variable for each customer, and, possibly, *iv)* compatibilities between customers and vehicles types . Each customer can be visited more than once, hence this is a "split delivery" problem. There are 7 types of vehicles available. Each type is characterised by *a)* a different number of compartments for which a maximum capacity for each fuel type is given (each fuel type can have a different density and a compartment can be filled by only one product at a time), *b)* a fixed annual cost, *c)* a variable operation cost, comprising a per-unit time cost (representing, e.g., salary), and *d)* a per-unit distance cost (representing, e.g., fuel cost). Each vehicle can perform one or more trips which are constrained to be in a certain time windows (morning and afternoon). During a working day, a vehicle can return to the depot to refill (i.e., the problem also features vehicle re-use).

Available Data We have at our disposal one month of data for a distribution centre (DC). To better analyse the algorithm, we modified the data to create two instances. Since a lot of customers require service on several days, it is possible to perturb the history to create more days. This results in 2 different multi-day instances, which we denote by DC1 and DC2, each of 25 days, with the number of customers each day varying in the range $[17, 52]$. Solving a single multi-day FSM across the whole

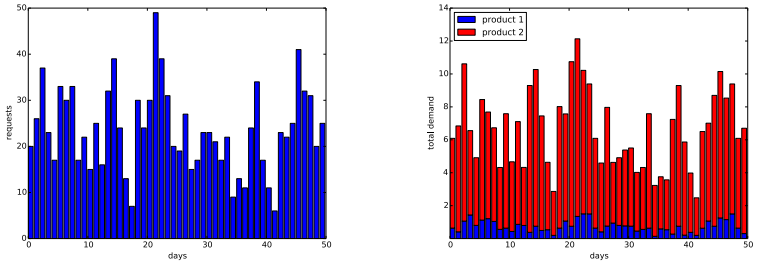


Figure 4.2: On the left we plot the number of requests for each day. On the right, the total demand for each day for the commodities (the y-scale is in hundreds of thousands). Note that we are plotting both instances next to each other, hence the horizon includes all the 50 days.

horizon would involve solving problems with more than 600 customers. In order not to compromise the data, we tried to maintain the original variation of total demand and the number of customers per day. Figure 4.2 gives a visual representation of how the daily scenario varies across the planning horizon. We did not have available data for hired vehicles. Therefore, we created two different scenarios for each instance, modelling a situation where hiring is cheap and one in which it is expensive. In the first, labelled "low", we set $h_t = \frac{3}{2}b_t$ and we retain the original per-unit distance and time cost equal. In the second, labelled "high", we set $h_t = 2b_t$ and we set the per-unit distance cost of the hired vehicles to 1.7 of the owned vehicles' one. In both scenarios the hiring periods are the single days. Summarising, we now have 4 difference instances, DC1-low, DC1-high, DC2-low, DC2-high. There is no prior fleet in the instances.

FSM-solver A detailed description of the problem can be found in Urli and Kilby [2017]. The authors solve it by means of a Large Neighbourhood Search (Shaw [1998]; Ropke and Pisinger [2006]) heuristic where the "repair" phase is implemented through Constraint Programming (Rossi et al. [2006]). Their heuristic was developed to support fleet size and mix and it is the FSM-solver we use to solve the sub-problem. Their method supports also a multi-day scenario. However, in Urli and Kilby [2017], the authors test it on a set of much smaller instances, pointing out that multi-day FSM scenarios are quite hard to solve. The instances considered here are significantly larger in size. We attempted to apply the solver on a multi-day scenario, but, already when considering few days, the solver struggles to find the first feasible solution and the quality of the final solution is quite bad. We do not go into details of the FSM-solver implementation; we refer the interested reader to Urli and Kilby [2017], where a better analysis of the FSM-solver is proposed.

R-solver In order to be able to solve the sub-problem in the RG method we modify the FSM-solver to include profits for each customer. This is a straightforward modification since the only differences are in how the objective is computed and in the removal of the constraint enforcing satisfying completely every customer. We decided to opt for an easy modification of the FSM-solver rather than developing a labelling algorithm for two main reasons: first, we are not trying to solve the problem to optimality. In column generation applications to routing problems heuristics are often used in order to generate columns faster. Labelling algorithms are invoked only when these methods fail and one needs to prove that there are no more columns with negative reduced costs, therefore optimality of the master is reached. A second reason, as we mentioned before, is the difficulty in defining a good dominance rule for the extensions of paths in the labelling algorithms. Having multiple commodities, multiple compartments which are not specifically dedicated to given commodity and considering split deliveries makes the definition of such rule very hard and labelling algorithms not a viable option.

Time Limit and Other Parameters In order to have a fair comparison we run FG, RG and RM in single-threaded mode with a time limit of 4 hours. In the following, we do not report computing times of the algorithm. This is because in all of the experiments, the computational time is close to the time limit, and the differences are significant. All the algorithms were coded in Python 2.7.10. The FSM-solver described in Urli and Kilby (2017) was coded in C++ and called as an eternal executable. We used the Python API of Gurobi 6.5 (Gurobi Optimization (2015)) to solve all models. Since the FSM-solver is stochastic, in order to have a robust result, we run each method 5 times on each instance and then take the average of the different runs. This is done for all the experiments.

4.7.1 Approach Validation

We first aim at validating the algorithms proposed, and, more generally, the idea of applying CG to a decomposition of the problem. In this section, we consider a simplified version of the problem where we are only allowed to buy vehicles, i.e. $E_t = H_{tp} = 0$ for all $t \in T, p \in P$. This is needed in order to compare with other methods that were developed for this particular case. We do not expect this to affect run times to a great extent.

Ideally, we would be able to compare the proposed approaches with other methods from the literature. We used the method proposed in Kilby and Urli [2016] to produce a fleet based on the days in the first "Pareto front". However, due to incom-

patibility constraints, this fleet was infeasible for many days in the test data. Since Kilby and Urli [2016] does not offer a method to overcome this limitation, we are unable to compare its performance.

Instead, we compare the performance of the proposed methods to other two standard approaches. The first, naive, approach, described in Urli and Kilby [2017], consists in solving each day separately with amortised fixed costs, i.e., $b_t/|D|$ (the overall fixed costs of the vehicles divided by the number of working days), then taking the fleet that is effectively the union of the fleets of vehicles obtained by solving each day separately. In other words, if $[F_d^t \mid t \in T]$ is the fleet obtained for day $d \in D$, the overall fleet is defined as $F_t = \max_d F_d^t$, $\forall t \in T$. The authors name it Union fleet (UF) method. We remark that the UF approach treats every day as a separate entity and therefore is clearly sub-optimal. However this is the approach that is used in the literature to solve multi-day FSM problems (see for example Yoshizaki et al. [2009]; Kopfer and Wang [2009]).

The second approach, that attempts to replace the approach of Kilby and Urli [2016], is to formulate a "big day" scenario. The logic here, often employed in practice, is to form a fleet that can meet the demand on the day with largest demand, and hence will be able to meet the demand on all other days. The method, which we name Subset Algorithm (SA), is illustrated in Algorithm 4. This is based on the fact that the FSM-solver described in Urli and Kilby [2017] supports the solution of FSM problems over several days.

Algorithm 4 Subset Algorithm

1: Fix $m \geq 1$
2: Rank the days based on the total load to be delivered on the day, in decreasing order.
3: Consider the first m days in the rank and denote them by $d_1, \ldots, d_m$.
4: Set the fixed cost of type t to $\frac{b_t m}{|D|}$ for all $t \in T$
5: Solve the FSM problem for the first m days and name the so obtained fleet F,
6: **for** $d \in D$ such that $d \notin \{d_1, \ldots, d_m\}$ **do**
7: adjust F to accommodate vehicles' compatibilities of day d
8: Solve d using F
9: **if** d is infeasible with F **then**
10: Set the routing cost of d to infinity
11: **end if**
12: **end for**
13: Return $z(m) = bF + \sum_{d \in D}$(routing cost on d)

The SA solves a "restricted" fleet design problem on the first m biggest days using amortised fixed costs, obtaining a fleet F. Then, it proceeds to solve the routing problem on all the other days. Note that, for a fixed day $d \notin \{d_1, \ldots, d_m\}$, the fleet F

is not guaranteed to be feasible. In an attempt to try to prevent infeasibility, we adjust F in Step 7. This is done only if vehicles' compatibilities imply we need to add some vehicles to F. As an example: if F has no vehicle of type $\bar{t}$ but there are customers (in day d) which can only be served by vehicles of type $\bar{t}$, we add the minimum number of vehicles of type $\bar{t}$ needed by F. This might not be enough to prevent infeasibility, as for example the placement and time windows of the customers could make a day infeasible even for the adjusted fleet, however there is no trivial way to prevent this possibility. If there is one infeasible day with the given fleet F we simply set the routing cost to infinity. The fixed costs used to solve the restricted fleet design are normalised (Step 4). However, the original vector b is used to compute the fixed costs of the final solution (Step 13). Note that there is a trade-off in increasing the number of considered days m. For small values of m, the produced fleet might be too sub-optimal and probably not feasible on some other day. For higher values of m, the fleet has a higher degree of guaranteed coverage, though Step 5 becomes computationally challenging. We run the algorithm SA with $m = 1, \ldots, 5$ to give SA a good chance of finding the best possible results it can reach. This way we enable a fair comparison. However, except for one run, the best results were found with $m = 3$.

For the UF method, we run the FSM-solver on each day for 20 minutes, which is enough to reach convergence. This amounts to a total of 500 minutes. We run the SA algorithm, for a fixed m, allowing $20m$ minutes to the multi-day FSM-problem solve in Step 5 and 20 minutes to every other day. Finally, we only take the best results of the runs with different values for m. Clearly, we are giving UF and SA an advantage in terms of computational time. Despite this advantage, the results will show that our methods outperform both UF and SA.

Given the computational difficulty of the problem, we do not have a lower bound available with which to compare our results. Moreover, we do not have the current solution adopted by the company. However, we can compute an "approximate lower bound" that will serve as a lower bound for comparison purposes. This is calculated assuming we can change the fleet each day. While this bound still relies on the accuracy of the heuristic technique, it does provide a conservative bound on the best possible solution that can be obtained using the given FSM solver. It therefore highlights the efficacy of the different approaches in choosing a fleet for the overall problem. We lower bound the operational and fixed costs separately. As per the operational cost, we run the FSM-solver, with fixed costs set to zero, 5 times on each day for 20 minutes (each run) and take the best run for each day. By summing the best results of all days, we obtain an approximate lower bound for the operational cost. We then compute a (formal) lower bound on the fixed costs as follows. Assume we have only one commodity, given a day d, any routing plan $s \in X_d$ must satisfy

the constraint

$$\sum_{t \in T} Q_t F_{ds}^t \geq TD_d$$

where Q_t is the capacity of vehicle type t and TD_d is the total demand of day d. Therefore, the fixed costs must satisfy the following inequality:

$$\sum_{t \in T} F_t^+ b_t \geq \min \left\{ \sum_{t \in T} b_t F_t^+ : \sum_{t \in T} Q_t F_t^+ \geq \max_d TD_d,\ F_t^+ \geq 0 \, \forall t \in T \right\}.$$

The right hand side is readily computable, thus we have obtained a lower bound for the fixed costs. Note that, if the underlying VRP has more than one commodity, the lower bound can be improved by adding a constraint for each commodity. This is also the case for other constraints, such as compartments or compatibilities between products and vehicles. The sum of the lower bounds on operational and fixed costs is taken as the overall approximated lower bound.

instance	method	cost	σ	operational	fixed	veh	idle	gap (%)
	UF	785727	27030	440847	344880	30.4	18.7	43.1
	SA	697739	18297	413939	283800	24.6	10.7	27
DC 1	FG	605781	9866	420141	185640	16.2	5.1	10.3
	RG	615971	2756	421271	194700	17.1	5.48	12.1
	RM	572783	2895	414083	158700	14.1	4.28	4.3
	UF	613004	9470	364214	248790	22.2	14.0	34.2
	SA	588488	9250	342458	246030	21.4	9.5	28.8
DC 2	FG	499821	5727	348892	150930	13.4	4.4	9.4
	RG	505045	3212	349195	155850	14.5	4.8	10.5
	RM	484004	3860	346754	137250	12.9	4.1	5.9

Table 4.1: Comparison of all methods. In these runs we do not consider the possibility of hiring vehicles.

We ran each method on both instances DC1 and DC2[5]. We recall that we are not considering hired vehicles in this section. In Table 4.1, we report the average overall cost (cost) across the 5 runs, its standard deviation (σ), a breakdown of the cost in its components (operational and fixed costs), the average number of vehicles (veh) in the fleet and the average number of idle vehicles per day (idle). Lastly, we report the average gap with respect to the approximated lower bound. The costs are approximated to the nearest integer.

It is easy to see that all the three proposed methods outperform both UF and SA.

[5]As previously mentioned, each method is run 5 times with its associated time limit.

Moreover, they are far more stable, as showed by the lower values of σ. As expected, the main saving is in the fixed costs since both UF and SA yield bigger fleets with a higher average of idle vehicles per day.

A particularly good result is the fact that our methods are able to produce solutions within 5% of the approximated lower bound. We stress that the lower bound is computed by taking the best operational cost each day (i.e., each day the fleet used to obtain such cost can be different), and a fleet satisfying only the capacity constraints. This particularly highlights the efficacy of the approach described.

Moreover, not surprisingly, we observe that RM always produces the best result. A further comparison among FG, RG and RM is presented in next section.

It is also interesting to see that the average number of idle vehicles is even further reduced, in proportion, than the average number of vehicles. When designing a fleet, a key indicator used by fleet managers is the vehicle utilisation. A rule of thumb is to keep the utilisation rate higher than a target percentage. Our approach does not consider this criterion explicitly in the objective. However, it is clear from Table 4.2 that the reduction of idle vehicles is a consequence of the efficiency of the method.

4.7.2 Methods Comparison

We now wish to compare the three proposed methods when hired vehicles are included. We run FG, RG and RM on the instances including hired vehicles. In Table 4.2, we report the results. The column labelled "hired" reports the average number of hired vehicles per day. Note that, when computing the average number of idle vehicles per day, only owned vehicles are considered.

A few remarks are in order. It is evident that introducing the possibility of hiring make the problem computationally harder. The solutions obtained by FG and RG on DC1-high, and the one obtained by FG on DC1-low, are worse, in terms of costs, than the ones the same methods obtained for DC1, i.e., when no hiring is allowed. However, we note that, despite the higher costs, the number of vehicles in the fleet and the number of idle vehicles are reduced significantly. It is easy to observe that RM outperforms the other methods consistently. The difference is made in the fixed costs, even though the operational cost is reduced as well. All three methods obtain fleets of similar size, but the composition and the use of hired vehicles changes, significantly influencing the fixed costs. RG performs better than FG on the *low* instances, and vice versa, FG performs better than RG on the *high* ones. However, by looking at σ and the average idle vehicles, RG seems more reliable than FG. Notably, both RG and RM have a very high utilisation rate, i.e., the average of idle vehicles per day is kept extremely low.

instance	method	cost	σ	operational	fixed	veh	idle	hired
DC1-high	FG	628786	8699	436822	191964	15	3.83	0.89
	RG	665113	1444	435942	229171	14.2	2.75	2.96
	RM	581638	12403	423868	157770	13	2.1	0.4
DC1-low	FG	629758	3793	440281	189477	11	2.62	3.62
	RG	562835	2547	422988	139847	7.2	0.8	3.34
	RM	548200	6859	417537	130663	8.2	0.57	2.1
DC2-high	FG	492968	9472	359067	133900	9.6	1.95	1.05
	RG	518074	581	355350	162724	10	1.93	2.1
	RM	475731	3327	355225	120506	10	1.54	0.31
DC2-low	FG	492510	6766	357251	135259	5.8	1.64	4.02
	RG	458698	1811	347481	111218	5.4	0.38	2.94
	RM	451361	1969	346481	104880	5.6	0.25	2.42

Table 4.2

4.7.3 Importance of Hiring Option

We now analyse the importance of the possibility of hiring vehicles and how this can help reducing the costs. In order to do so we compare the results on both instances DC1 and DC2 with and without the possibilities of hiring vehicles. Since RM is the most performing of the algorithms we proposed, for these experiments, we use only algorithm RM. In Table 4.3, we report the results. In the instances DC1 and DC2, hiring is not allowed.

instance	cost	σ	operational	fixed	veh	idle	hired
DC1	572783	2895	414083	158700	14.1	4.28	-
DC1-low	548200	6859	417537	130663	8.2	0.57	2.1
DC1-high	581638	12403	423868	157770	13	2.1	0.4
DC2	484004	3860	346754	137250	12.6	4.1	-
DC2-low	451361	1969	346481	104880	5.6	0.25	2.42
DC2-high	475731	3327	355225	120506	10	1.54	0.31

Table 4.3: The effect of hiring vehicles. RM algorithm

It is evident that introducing the possibility of hiring vehicles has a tremendous impact on the fleet composition and utilisation, regardless of the hiring being more or less expensive (high/low scenarios). The fixed costs are reduced without significantly changing the operational costs. Moreover, the size of fleet is reduced and the number of idle vehicles is decreased substantially.

4.7.4 Impact of Route-based Model

We want to analyse the advantage of introducing a second phase in RM. Let us recall that RM is split in two phases. In the first, we use model **F** to generate a good set of fleets. In the second, we initialise model **R** with all routes generated in the first phase, and proceed to solve it by means of CG. Note that both model **F** and **R** are standard Set Covering problems.

During FG, the routes are recorded, though they are not explicit in model **F**. In the transition from the first to second phase, the recorded routes are used as the initial route set. In the second phase, we generate new routes through the solution of the sub-problems. Note that, in this second phase, we are actually solving a restricted version of model **LR**. At the end of the second phase we solve the integer model **R**. In this section, the focus is on the advantage of transitioning from a fleet-based model to route-based one. Therefore, after the first phase, we initialise and solve the integer model **R**, directly, and name this step *route-reuse*. By looking at how much the various statistics are reduced by this single step, we obtain an insight on how useful and efficient is the transition from fleet-based model **F** to route-based model **R**. In Table 4.4, we report the percentage change of some statistics (the total, operational and fixed costs, as well as fleet size, average of idle vehicles per day) due to the route-reuse step. For example, referring to instance DC1-high, when reusing the routes generated in the first phase of RM, the total cost decreases by 6.35% while the operational cost increases by 2.04%.

instance	Δ-cost	Δ-operational	Δ-fixed	Δ-veh	Δ-idle
DC1-high	-6.35	2.04	-17.58	-26.88	-63.85
DC1-low	-10.25	0.40	-27.37	-54.26	-92.02
DC2-high	-5.6	2.5	-19.3	-27.4	-66.08
DC2-low	-8.68	1.31	-26.03	-56.76	-92.822

Table 4.4: The effect of the route-reuse step. For each statistic, we report the percentage decrease we observe after solving the associated **R** model. All numbers in the table are percentages.

We can observe a consistent behaviour. The total and fixed costs are reduced considerably, sometimes at the expenses of the operational cost. Moreover, the fleet size and the number of idle vehicles are dramatically reduced. Note that in the first phase of RM we do not consider hired vehicles.

To make a stronger point, we apply the same idea to the FG method. We take all the routes generated by the method and solve the associated **R** model only once, to test the effect of reuse of the routes. The difference here is in the fact that FG considers hired vehicles, while the first phase of RM does not. The percentage changes due to

the route-reuse step are reported in Table 4.5. In addition to the previous table, we report the change of the average of hired vehicles per day. The results show the same behaviour highlighted before. The decrease of the various statistics is lower. For the *high* instances, this is due to the fact that when considering hired vehicles, FG performs worse, and the pool of routes generated might not be as good as the one generated when FG does not consider hired vehicles. Conversely, for the *low* instances, considering hired vehicles allows FG for a better optimisation of the fixed costs than the first phase of RM. In both cases, the route-reuse step makes a better use of the hired vehicles. Accordingly, all statistics, including the operational cost, are improved.

We can conclude that the transition from fleet-based model to route-based model is very beneficial. This is also a consequence of how effective the first part is in generating a good set of routes. The combination of the two phases proves to work efficiently.

instance	Δ-cost	Δ-operational	Δ-fixed	Δ-veh	Δ-idle	Δ-hired
DC1-high	-2.44	0.45	-9.02	-16	-50.39	46.59
DC1-low	-3.85	-12.7	-0.04	-16.36	-65.7	-10.04
DC2-high	-1.08	-4.63	0.24	-2.08	-27.69	-15.08
DC2-low	-4.36	-14.86	-0.37	-10.34	-83.55	-18.74

Table 4.5: The effect of the route-reuse step when hiring is included in the first phase.

4.8 Contributions and Conclusion

Strategic and tactical-level decisions in the context of transportation can have a significant impact on the day-to-day operations, and can make the difference between running a profitable business or not.

In this chapter, we analysed the problem of designing a fleet of vehicles for a long time horizon. We considered both hired and owned vehicles. The single-day version of our problem can be seen as the Integrated Operational Transportation Problem. In the literature fleet design over a long horizon has been widely overlooked. In this chapter, we proposed a general approach to tackle the problem. Our approach is independent of the particular routing problem considered on a single day and leverages the power of a FSM-solver and, possibly, a shortest path problem solver, each of which can be chosen by the user.

We proposed three column-generation based heuristic methods, and compare their performance on a real-world instance. We showed how the method combining column-generation at two levels – fleet and route – outperforms the others. We

compared the results obtained with, and without, the possibility of hiring external vehicles. The simulations show that considering the option of hiring extra vehicles can be beneficial, and reduce the overall costs significantly. We also discussed how other subcontracting options can be added to the problem without major change. To the best of our knowledge, very few papers consider fleet design at a tactical level, and no published paper considers tactical fleet design problems with the possibility of hiring vehicles. Moreover, papers considering FSM problems with external subcontracting focus on single day problems, i.e. they focus on the operation level of the problem, and simple versions of the underlying routing problems.

We believe that the problem as formulated in this section is quite general. The multi-day nature of the problem, the possibility of considering rich version of the underlying VRP and the possibility of considering several subcontracting options make our models realistic and applicable. In particular, including the possibility of hiring extra vehicles, as proven by the computational experiments, brings great flexibility to the fleet design process.

Future work could involve the inclusion in the model of inter-day constraints, such as consistency in service over the horizon or scheduling decision. However, these are problem-dependent and less general than the features already considered. The two papers on which this chapter is based (see Chapter 1) conclude our work on fleet design. Nevertheless, the two inter-day constraints mentioned above are the topic of the next two chapters.

Chapter 5

Tactical Routing Strategy: Splitting Deliveries

5.1 Introduction

In the CVRP, we wish to provide goods or a service to a set of customers using a known fleet. The planning horizon is usually a single day. The quantities to be delivered to the set of customers is given as input, and the goal is to find a set of routes that minimises the routing cost. As introduced in Section 2.3, part of the literature is dedicated to variants considering a longer planning horizon that can comprise several days[1]. In practice, there are situations where a certain flexibility on the delivery dates is possible. For example, it might be convenient for the transport company to partially fulfil an order on the requested day (in order to ensure the customer has sufficient stock) but return the next day to complete fulfilment. In these cases it is of interest to investigate the possible savings that could come from considering the delivery dates, and possibly the quantities delivered, as additional decision variables.

Situations where there is total freedom on when and how much to deliver, within constraints on customers' inventory level, are usually modelled by the class of Inventory Routing Problems (IRPs). On the other hand, problems where we can choose the frequency of visits but not the quantity to be delivered (which could be fixed or may not even be relevant) belong to the class of Periodic Vehicle Routing Problems (PVRPs). We will show that our approach falls in some way between these two well-studied problems.

IRPs and PVRPs are well known and have received considerable attention by researchers in the past years. However, it is not always possible to combine the inventory management and routing aspects. The responsible actors for transportation,

[1] As stated in Section 2.3 the term "day" should be understood to indicate a period, i.e.,"a convenient unit of time".

and for inventory management, might not coincide. Communication between the two actors may be hard, due to difficulties in data sharing. Or it may be that the actors would not benefit equally from an integrated system. The result is that in practice, routing and inventory management are often optimised independently.

The real-world problems that motivate this study arise in long-haul grocery delivery, and fuel delivery problems. In these contexts, customers place orders up to two weeks in advance, nominating a delivery quantity, and a desired delivery date. A transport company fulfils these orders, but does not have access to rates of demand for the product, nor the customer's capacity to hold goods. Now consider the following: we have customers A and B, located relatively close together. Vehicles do not have sufficient capacity to visit both A and B together, but could *partially* fulfil A after visiting B. The remainder of customer A's demand could then be delivered the next day. So long as the amount initially delivered to A was sufficient to avoid a stock-out, such a solution may lead to reduced costs for the transport company (which could then be partially returned to the customers, in order to compensate for inconvenience). The motivation for this study was to look at how much such a solution may be able to save.

In both our motivating problems, customers need to be visited several times in a week, if not all days. If a customer is already accepting split deliveries on a single day, the idea of splitting a request across consecutive days would appear viable. Indeed, in many applications, e.g. fuel delivery, it is unlikely that the whole amount requested is consumed within a day, although it is required that a minimum percentage is delivered on the day of request so that stock-outs are avoided. Some companies already implement this strategy. If customers require service on consecutive days, splitting a delivery would not even imply more visits than already scheduled, but only an adjustment in the quantities delivered. Moreover, if the customer locations exhibit a degree of clustering – as we observed in our problem – inserting an additional customer visit in a day plan, would not be costly, and might actually reduce the overall routing cost. Following the idea presented in Gulczynski et al. [2010] we implement constraints on minimum delivery amounts. Namely, both a single delivery to a customer, and the total delivery on the day of request, have to be greater than fixed percentages of the customer's demand. The addition of these constraints stems from practical considerations. Given that a visit has an intrinsic cost (due to paperwork, service time) it is not practical to visit a customer many times. Moreover, as said before, in order to avoid stock-outs, we impose the constraint that a certain proportion is delivered on the day the order was requested. We name this problem Multi Day Split Deliveries VRP (MDSDVRP).

The MDSDVRP can be seen as an extension of the Split Deliveries VRP (SDVRP).

A multi-day planning horizon might be reformulated as an SDVRP with disjoint time windows on requests (representing the different delivery days); but this would not be able to represent the shifting of demand across days. Alternatively, one could introduce several copies of the same vehicles and add compatibility constraints to model the multi-day structure. However, this would again not capture the ability to shift demand across days. The possibility of splitting a delivery over days introduces a new degree of complexity not present in the SDVRP. In particular, this difference is reflected in the fact that some properties proven to be valid for optimal solutions of SDVRP do not hold for the MDSDVRP, as we show in Section 5.5.

The contribution of this chapter is fourfold: first, we propose a new problem model that integrates inventory management and vehicle routing. We consider some well-known theoretical properties of optimal solutions of SDVRPs and we extend them, or show they are no longer valid, to optimal solutions of MDSDVRPs. We then propose a mathematical programming formulation for the problem, together with a simple heuristic that is used to warm-start the solution process. Finally, we present an extensive experimental analysis of the proposed model aimed at quantifying the savings introduced by allowing split deliveries over days. We also focus on the impact that different features of an instance have on the possible savings.

The rest of the chapter is organised as follows: in Section 5.2 we briefly explain the reason the led us to focus on this particular problem. In Section 5.3, we review the literature related to the MDSDVRP. The problem is formally introduced in Section 5.4. In Section 5.5, we extend the theoretical properties valid for the SDVRP. A mathematical formulation for the problem and the solution method are proposed in Section 5.6. In Section 5.7, we present our experimental analysis. We give some conclusions in Section 5.8.

5.2 Motivation

In the problem presented in the previous chapter, as well as in the problem considered in Kilby and Urli [2016], customers may have several associated requests on different days throughout the horizon. Some of these requests are already split over days by the agents in play: transportation providers and the customers. However, this process is not optimised. Even if a centralised control of all customers' inventory level is a viable option, the approach proposed in this chapter may be far more applicable and easy to implement, since the splitting of requests over days already happens. In this chapter, we propose a first, theoretical and experimental analysis of such approach to a simple VRP.

5.3 Related Work

The problem studied in this work is closely related to the one proposed in Archetti et al. [2015b]. Here the authors study a similar problem, that they name Multi Period VRP with Due dates (MVRPD). Namely, they consider the possibility of choosing the delivery date for each customer within a window of a few days. However the delivery cannot be split between different vehicles. An inventory cost for holding the products in a central depot is considered. They analyse the impact this flexibility has on the routing cost, showing that it can reduce it substantially. Their work is at a theoretical level and aims at computationally exploring the advantages such a strategy presents.

As already mentioned, two broadly related areas of research are the IRP and the PVRP. These two classes of problems were already introduced in Section 2.3. Therefore, here we only highlight the differences with the MDSDVRP. The IRP is more general, in that we have complete freedom on the choice of when and how often to serve customers. The problem with this approach is that it might have limited application in real world scenario, as it requires complete control over the customers' inventory level, and also some form of knowledge of the rates of consumption. However, if such a deep integration of the inventory management and routing aspects is feasible, the possible savings are substantial (Andersson et al. [2010]). In Andersson et al. [2010] the authors review the status of research on problems aiming at combining inventory management and routing, highlighting the gaps between industry and research. The authors identify the conditions under which an integration can be feasible and beneficial. However, they note that in industry, the inventory and routing aspects are still often treated separately. The IRP constitutes a very challenging problem, even beyond the problems of application mentioned here.

The PVRP models the situation where customers request a certain frequency of service, with the flexibility of choosing the precise days of service. Unlike the problem studied here, the customers determine the service frequency and the quantities to be delivered. For a review of the IRP and PVRP, see Section 2.3.

In terms of flexibility, the MDSDVRP lies between the IRP and PVRP. The freedom of scheduling decisions allowed in the MDSDVRP is not as general as in the IRP, and clearly different from the decisions considered in the PVRP. In the IRP, the focus is on minimising the routing and inventory cost while assuring there is no stock-out at any customer at any time. In the MDSDVRP, the inventory costs are ignored, and there is more flexibility in the routing: we only require that a certain percentage of the demand be delivered by the day of the request. The situations modelled in the MDSDVRP and the PVRP clearly differ. The clearest difference is that in the latter

there is no possibility of deciding the portion of demand to be delivered.
Two other key aspects of our formulation are:

- we consider splitting deliveries on a single day;
- we include, optionally, a fleet size aspect in the problem.

We consider, in parallel to the standard case, a fleet size variant of the problem. The literature on FSM problems in reviewed in Chapter 4. Since we consider a homogeneous fleet, our problem is a special case of the FSM. However, this is enough to show that the strategy proposed has an impact on the fleet.

Finally, we briefly review the literature on split delivery routing. In the SDVRP, a customer can be visited by more than one vehicle. Hence the SDVRP is a generalisation of the classical CVRP. It was first presented in Dror and Trudeau [1989] and since then it has received a lot of attention in the research community. An extensive survey presenting exact methods, heuristics and applications can be found in Archetti and Speranza [2012]. We refer the reader to references therein. It has been shown that allowing split deliveries can lead to a reduction of up to 50% of the routing cost. This and other properties are presented and proven in Archetti et al. [2006a]. In Section 5.5, we extend these properties, or show they no longer hold, to the MDSDVRP. As already mentioned, the main difference between the SDVRP and the MDSDVRP is the multi-day structure. We consider a particular version of the SDVRP named Split Deliveries VRP with Minimum Amount constraints (SDVRP-MA), which was introduced in Gulczynski et al. [2010]. This formulation adds constraints to the SDVRP requiring that a vehicle must delivery at least a minimum percentage of the demand if it visits a customer. This seeks to avoid excessively splitting a customer's request, creating visits that deliver very small amounts of demand, as this is not desirable in real applications.

5.4 Problem Formulation

We consider a multi-day planning horizon. We denote by $\overline{D}$ the number of days and by $D = \{1, \ldots, \overline{D}\}$ the planning horizon. We are given N customers, each of which requires service on a (different) subset of days of D. We denote the set of requests by C. Hereinafter we identify customers with requests. Therefore two customers in C might share the same location, as they are originated from two different requests of the same original customer. We denote by q_i the demand of customer $i \in C$. The depot is denoted by 0. Travelling between two customers i and j incurs in cost c_{ij}. We assume that the distance matrix is symmetric and satisfies the triangle inequality.

The assumption on symmetry can be relaxed in a straight-forward way. We have available an unlimited number of vehicles of capacity Q. There is a fixed cost b for using a vehicle. That is, we pay b if we use a vehicle once. The overall fleet is the maximum number of vehicles used in a single day. We will consider both the case $b = 0$, where the focus is on the routing cost only, and $b > 0$, where the goal is to strike an optimal balance between fixed and routing costs. The delivery to a customer can be split between different vehicles on a single day. As previously mentioned, we impose a minimum delivery quantity: if a visit is made to a customer, then a minimum fraction $m_a \in [0, 1]$ of its demand has to be delivered. We do not impose granularity on the demand, i.e. any fraction of the demand can be delivered. There are situations where the demands are composed of a number of indivisible items (e.g. bottles), and we will see in Section 5.6 that these situations reveal another difference between SVRP and MDSDVRP.

We allow the demand to be delivered partially on the day of request and partially on the next α days. We refer to α as the *splitting horizon*. We assume $0 \leq \alpha < \overline{D}$. For example, if $\alpha = 1$, a customer's demand can be split over the original day of request and the next day. We chose to model the problem in a cyclical fashion, i.e. the *next* day of day $\overline{D}$ is day 1. For example, if $\alpha = 1$, a request due on day $\overline{D}$ can be split and partially delivered on day 1. This is because we are trying to minimise the fleet as well as the routing cost. Therefore, we cannot introduce a penalty for delivering the demand of a customer on day $\overline{D}$ to the next planning horizon, as is done in Archetti et al. [2015b]. This, in fact, would decrease the total demand delivered and impact the necessary fleet by hiding part of the demand. Indeed, we could postpone the delivery of part of the demand of the last day, but if the horizon is not seen as a cycle, this demand would not be considered in the problem. On the other hand, forbidding to move the demand on the last day goes against the idea of adjusting the schedule and it does not reflect the fact that in real applications we have a sequence of planning horizon and postponing part of the last day's demand is allowed. As an example, consider an instance of two days, with two customers, one per day, with the same location and requesting $Q + 1$. If the problem is cyclical, the size of the fleet has to be 2. However, if we can postpone the demand of the second day without having to serve all customers, the optimal fleet size becomes 1. Hence, in the following, a day index d will be intended as d mod $\overline{D}$.

Let us introduce some notation that will be used throughout the rest of the chapter. We denote with A the set $\{0, \ldots, \alpha\}$. For all $i \in C$, the original day of request of i is denoted with d_i. Moreover, for $s \in A$, we set $d_i^s = d_i + s$ and define the set $D(i) = \{d_i^s \mid s \in A\}$. Therefore $d_i^0 = d_i$ and $D(i)$ is the set of days when i can be visited. Given a day $d \in D$, we write $C_d = \{i \in C \mid d_i^0 = d\}$, the set of customers

whose original day of request is d. The set of all the possible customers that can be visited on day d is denoted by P_d. This can be defined as follows

$$P_d = \bigcup_{s=0}^{\alpha} C_{d-s} = \{i \in C \mid d_i^s = d \text{ for some } s \in A\}.$$

Whenever we include the depot in a set, for example C, of customers we denote the extended set by $\overline{C}$.

We constrain a fixed portion $m_d \in [0,1]$ of a customer's request to be delivered the day of original demand. We also require that, if a customer is visited, then at least m_a of its demand, with $m_a \in [0,1]$, must be delivered. We refer to these sets of constraints as the *minimum delivery amount* constraints. Note that if $m_a = 1$, we cannot split the deliveries between vehicles. If $m_d = 1$ or $m_a > 1 - m_d$, we can only split deliveries within the same day. Therefore, from now on we will assume that $m_d < 1 \wedge m_a \leq 1 - m_d$. The problem described in Archetti et al. [2015b] corresponds to the case $m_a = 1, m_d = 0$ plus additional inventory costs.

The goal is to determine a number of vehicles F and a set of routes such that the total demand of each customer across the horizon is satisfied, the minimum delivery amount constraints are not violated, and the total cost (the sum of fixed and routing costs) is minimised.

5.5 Theoretical Properties

In Archetti et al. [2006a], the authors review some of the properties of the optimal solution of an SDVRP. The goal of this section is to extend those properties to the MDSDVRP. A first simple observation is that, given an optimal solution ϕ to the MDSDVRP, the subset of routes of ϕ performed on a single day d can be seen as an optimal solution of a SDVRP on d, where the demands are defined to agree with the quantities delivered by ϕ on d. Therefore, if $m_a = 0$, the subset of routes of ϕ on each single day d satisfies all of the properties characterising an optimal solution of an SDVRP.

5.5.1 k-split cycles

In Dror and Trudeau [1990], the authors introduced the following definition:

Definition 1. Given k customers $i_1, i_2, \ldots, i_k$, and k routes such that route 1 visits customers i_1 and i_2, route 2 visits customers i_2 and $i_3, \ldots$, route $k-1$ visits customers i_{k-1} and i_k and route k visits customers i_k and i_1. The subset of customers $i_1, i_2, \ldots, i_k$ is called a k-split cycle.

They also proved the following property:

Property 1 (Dror and Trudeau [1990])**.** *Given an instance of the SDVRP, if the cost matrix satisfies the triangle inequality, then there exists an optimal solution where there is no k-split cycle (for any k).*

Property 1 and its proof hold verbatim for the MDSDVRP if $m_a = m_d = 0$. The only difference is that now the routes can belong to different days. In Gulczynski et al. [2010], it was shown that this property does not hold if a minimum delivery amount for a single delivery is imposed. Therefore, if m_a or m_d are strictly positive, Property 1 is not valid for the MDSDVRP.

5.5.2 Number of splits and routes

Let r_i be the number of routes serving a customer i. We say the number of splits at customer i is $r_i - 1$. The total number of splits is $\sum_{i \in C}(r_i - 1)$. In Archetti et al. [2006a], the authors proved that, if the matrix satisfies the triangle inequality, there exists an optimal solution to the SDVRP where the number of splits is strictly less than the number of routes. Not only does this property not hold for the MDSDVRP – regardless whether m_a is zero or not – but we also have the following result:

Property 2. *If $m_d > 1/2$, the ratio between the total number of splits and number of routes of an optimal solution of a MDSDVRP instance is unbounded.*

Proof. Assume $D = \{1, 2\}$, $\alpha = 1$ and $m_d > 1/2$. Set $Q = 1$. Fix the number of customers N and assume that the first $N - 1$ customers request service on the first day and their demand is $q_i = \frac{1}{(N-1)m_d}, i = 1, \ldots, N-1$. The remaining customer, labelled by N, requests service on the second day, with $q_N = \epsilon > 0$, where ϵ will be defined later in the proof. Suppose that the distances between customers are $c_{ij} = \delta, c_{0i} = M$ for all $i, j \in C$ where $M \gg \delta$. The minimum total delivery on the first day equals $m_d \sum_{i=1}^{N-1} q_i = 1$. Therefore, we need at least one route to deliver the minimum necessary to each customer. We can deliver all the remaining quantities on the second day. Indeed, a simple calculation shows $(1 - m_d) \sum_{i=1}^{N-1} q_i + \epsilon = \frac{1-m_d}{m_d} + \epsilon$, and it is easy to see that, given that $m_d > 1/2$, this quantity is smaller than 1 if ϵ is small enough. Given that M is much bigger than δ, and that we need at least one route per day, an optimal solution must have only two routes, one delivering all the minimum allowed to each customer on the first day, one delivering all the rest in the second day. An illustration, for $N = 4$ on the example is given in Figure 5.1. Therefore, an optimal solution of this instance has 2 routes and $N - 1$ splits. Since N is arbitrary, we have proved the property. □

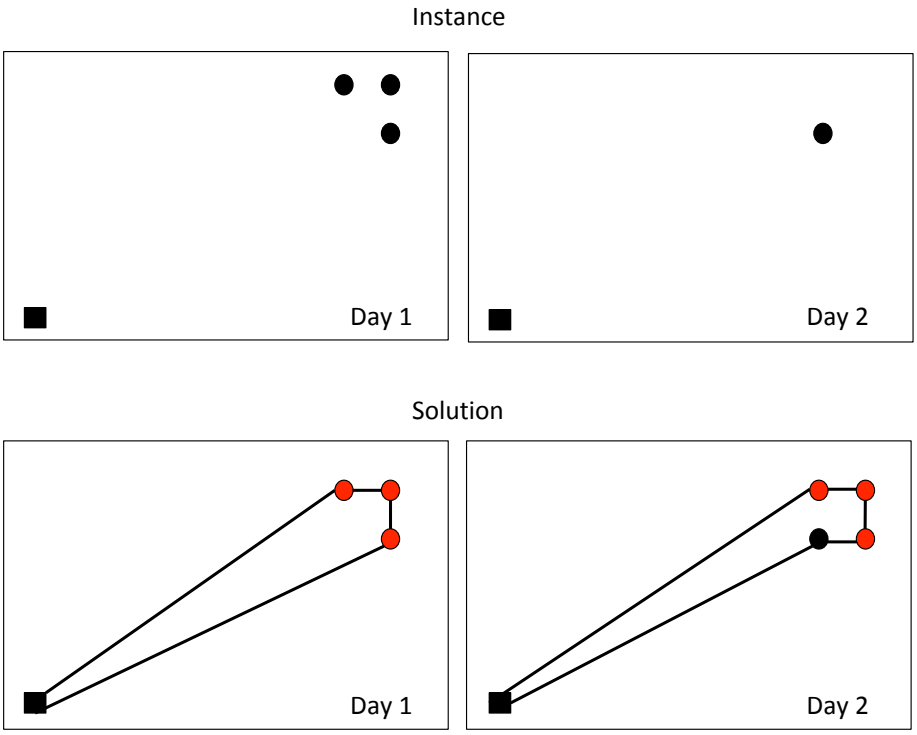


Figure 5.1: In the illustration, the square represents the depot and the circles the customers. In the top part, we plot the instance, in particular the original day of requests of the customers. In the bottom part the optimal solution is illustrated. The customers whose demand is split are in red. Note that the distances are not euclidean in this figure.

5.5.3 Bounds on the number of routes

Let us write $v_d = \lceil \sum_{i \in C_d} \frac{q_i}{Q} \rceil$ for any $d \in D$. If we consider the SDVRP instance induced by a day d, assuming delivery on the originally requested day, then v_d is the minimum number of vehicles needed to serve all customers. For the VRP the rounded-up term is replaced by the value of a bin packing problem.

For the MDSDVRP, the computation of a lower bound is slightly more involved as the days are not independent. Assume for now that $m_a = 0$. For all $s \in A$ and $d \in D$ we define the variables $m_d^s \in [0,1]$ to be the percentage of total demand moved from day d to day $d+s$. For these numbers to make sense, we have to impose, for every fixed $d \in D$, that

$$\sum_{s=0}^{\alpha} m_d^s = 1 \quad \wedge \quad m_d^0 \geq m_d.$$

Let us denote by q_d the total amount of demand delivered on day $d \in D$, this can be

computed as follows

$$q_d = \sum_{s=0}^{\alpha} (m_{d-s}^s \sum_{i \in C_{d-s}} q_i).$$

We define the set of all possible tuples that satisfy the minimum delivery amount constraints,

$$S = \{\{m_d^s\}_{d \in D, s \in A} \mid \sum_{s \in A} m_d^s = 1 \wedge m_d^0 \geq m_d \ \forall d \in D\}.$$

The minimum number of vehicles needed for the whole horizon is computed as follows:

$$\min_S \max_{d \in D} \Big\lceil \sum_{s \in A} m_{d-s}^s \sum_{i \in C_{d-s}} \frac{q_i}{Q} \Big\rceil = \min_S \max_{d \in D} \Big\lceil \frac{q_d}{Q} \Big\rceil. \tag{5.1}$$

In other words, for a fixed tuple in S, we use the formula for the SDVRP to compute the minimum number of vehicles for each day and then we take the maximum over all days. We then take the minimum over all possible tuples in S.

In the case $m_a > 0$, we cannot compute exactly the minimum number of vehicles. The formula (5.1) can be used to compute a lower bound on the minimum number of vehicles, if, for all $d \in D, s \in A$, we add the following condition to the definition of set S:

$$m_d^s \sum_{i \in C_d} q_i \geq m_a \min_{i \in C_d} q_i \quad \vee \quad m_d^s = 0.$$

This means that, the total demand moved from day d to day $d + s$ has either to be zero or to be larger or equal to the minimum delivery on day d.

Note that, for simplicity, we are working under the assumption that customers are identified with requests. If we removed this assumption, the computation of the minimum number of vehicles would be a considerably more complex problem. Indeed, the variables m_d^s would not be enough to capture the fact that, splitting a request on a day $d \in D$ may have an effect on the minimum delivery constraints on the next days (for example if a customer requests service on two consecutive days).

In Archetti et al. [2011] the authors prove that, provided the distances satisfy the triangle inequality, there exists an optimal solution to the SDVRP where the number of vehicles is not greater than $2v_d$. If we define,

$$\overline{v}_d = \lceil \sum_{i \in C_d} \frac{q_i}{Q} + (1 - m_d) \sum_{s=1}^{\alpha} \sum_{i \in C_{d-s}} \frac{q_i}{Q} \rceil$$

we have the following result for the MDSDVRP.

Property 3. *If the distances satisfy the triangle inequality, then there exists an optimal solution to the MDSDVRP where the number of vehicles used on day d is not greater than* $2\overline{v}_d$

Proof. The maximum quantity delivered on day d is

$$\sum_{i \in C_d} \frac{q_i}{Q} + (1 - m_d) \sum_{s=1}^{\alpha} \sum_{i \in C_{d-s}} \frac{q_i}{Q}.$$

By applying the aforementioned result in Archetti et al. [2006a] the property is proven.

□

5.5.4 Worst Case Analysis

In Archetti et al. [2006a], the authors present a detailed worst case analysis for the SDVRP when compared to the VRP. A worst-case analysis consists of a set of bounds and instances aimed at comparing the performance of VRP and SDVRP. Note that, in Archetti et al. [2006a], the authors do not consider a fixed cost for using a vehicle. In order to compare with their result, for the rest of this section we will assume that the fixed cost is zero ($b = 0$). In particular, they proved that savings generated by allowing split deliveries can be up to 50%. Formally, consider an instance of the VRP and let us denote by $z(\text{VRP})$ its cost and by $z(\text{SDVRP})$ the cost of the same instances where split deliveries are allowed.

Property 4 (Archetti et al. [2006a])**.** *It holds* $\frac{z(VRP)}{z(SDVRP)} \leq 2$ *and the bound is tight.*

Denoting with $z(\text{SDVRP-MDA}(m_a))$ the optimal cost for the same instance with minimum delivery amount constraints on single visits, in Gulczynski et al. [2010], the authors prove that, if $m_a = 2/j$ for some integer $j \geq 4$, a tight bound for the ratio $\frac{z(\text{VRP})}{z(\text{SDVRP-MDA}(m_a))}$ lies in the interval $[2 - m_a, 2]$, although a tight bound is not provided.

In a similar spirit, our analysis focuses on the benefits coming from splitting the demand over different days. We write $\text{MDSDVRP}(\overline{D}, \alpha, m_d, m_a)$ to explicitly represent the parameters associated with the multi-day structure of the problem: number of days $\overline{D}$, the splitting horizon α and minimum amount constraint parameters m_d and m_a. If $\alpha = 0$, we assume $m_d = 1$, even if, in this case, the value of m_d does not change the problem. Again, $z(\text{MDSDVRP}(\overline{D}, \alpha, m_d, m_a))$ denotes the cost of an optimal solution for $\text{MDSDVRP}(\overline{D}, \alpha, m_d, m_a)$.

Property 5. *Assume* $b = 0$*. It holds*

$$\gamma(\overline{D}, \alpha, m_d, m_a) := \frac{z(MDSDVRP(\overline{D}, 0, 1, m_a))}{z(MDSDVRP(\overline{D}, \alpha, m_d, m_a))} \leq \min\{\alpha + 1, \left\lceil \frac{1}{m_d} \right\rceil\}.$$

If $m_a < 1 - m_d$, *then* $\sup_{\overline{D}>0} \gamma(\overline{D}, \alpha, m_d, m_a) \geq 2$. *That is, for any* $\eta > 0$, *there exists a MDSDVRP instance with a sufficiently large horizon for which the benefit of splitting over* α *days is greater that* $2 - \eta$.

Proof. Consider a fixed number of days $\overline{D}$, a fixed integer $\alpha > 0$ and an optimal solution ϕ for MDSDVRP$(\overline{D}, \alpha, m_d, m_a)$. We first prove that $\gamma(\overline{D}, \alpha, m_d, m_a) \leq \alpha + 1$. Consider the following operation: given a route r, operated on a day d, we can create $\alpha + 1$ new, possibly empty, routes r_s as follows: for $s \in A$, route r_s contains all customers i visited by r such that $d = d_i^s$. Each route r_s is then moved to day $d - s$. If we apply the described operation to all the routes in the solution, we obtain a new solution which is feasible for MDSDVRP$(\overline{D}, 0, 1)$. Since the sum of the cost of routes originated by a route r is at most $\alpha + 1$ times the cost of r (due to the triangle inequality), the cost of the new solution increases by at most a factor of $\alpha + 1$. Hence, we have proved the bound.

We now prove $\gamma(h, \alpha, m_d, m_a) \leq \left\lceil \frac{1}{m_d} \right\rceil$, which completes the proof of the first statement in the property. Consider the solution ϕ', obtained from ϕ by applying the following operation: given a route r, operated on a day d, remove from r all customers i such that $d_i \neq d$, i.e., all customers whose original day of request is not d. Then, consider $\left\lceil \frac{1}{m_d} \right\rceil$ copies of the so obtained route. It follows that ϕ' is a solution to MDSDVRP$(\overline{D}, 0, 1, m_a)$ and satisfies $z(\phi') \leq \left\lceil \frac{1}{m_d} \right\rceil z(\phi)$. This proves the bound.

We now assume $m_a < 1 - m_d$ and prove, by building an example, the second claim. Note that, if $\overline{D} = 2$, then $\alpha = 1$ and the first claim of the property implies $\gamma(\overline{D}, 1, m_d, m_a) \leq 2$. Fix $\overline{D} > 2$. Consider $\overline{D}$ days and a single customer, located at a distance M from the depot, requiring service every day. We denote by q_d its demand on day $d \in D$. Let us assume, for now, that $m_d \geq m_a$. Define $\delta := \frac{1 - m_a - m_d}{\overline{D} - 2}$. The assumptions imply $1 > \delta > 0$. Assume $Q = 1$. Choose $0 < \epsilon < \min\left\{ \frac{m_a}{1 - m_a}, \frac{\delta}{1 - \delta}, \frac{m_d}{2 - m_d} \right\}$ and define $q_d = 1 + \epsilon$ for $d = 1, \ldots, \overline{D} - 1$ and $q_{\overline{D}} = \epsilon$. An optimal solution of MDSDVRP$(\overline{D}, 0, 1)$ has two out-and-back trips for the first $\overline{D} - 1$ days and only one trip for the last day. Its cost is $4M(\overline{D} - 1) + 2M$.

Let us define the following numbers

$$s_d = \begin{cases} 1 - m_a - (d-1)\delta & \text{for } d = 1, \ldots, \overline{D} - 1 \\ 1 & \text{for } d = \overline{D} \end{cases}$$

Note that $1 - m_a = s_1 > s_2 > \cdots > s_{\overline{D}-1} = m_d$. The number s_d represents the fraction of quantity q_d delivered on day d. We assume that the remaining quantity, $q_d(1 - s_d)$, is delivered on the next day. An illustration of this example is given in Figure 5.2. We now verify that this strategy yields a feasible solution with only one route a day. For

all $d = 1, \ldots, \overline{D} - 1$, we have $s_d \geq s_{\overline{D}-1} = m_d$ and $1 - s_d \geq 1 - s_1 = m_a$. Therefore, the minimum delivery amount constraints are satisfied. The capacity constraints can be stated as $(1 - s_d)q_d + s_{d+1}q_{d+1} \leq 1$, $\forall d$. A simple computation shows that these are satisfied thanks to the definition of ϵ, q_d, s_d. This solution is optimal, as we cannot use less than one route a day, and its cost is $2M\overline{D}$. Therefore, we have $\gamma(\overline{D}, \alpha, m_d, m_a) \geq \frac{4M(\overline{D}-1)+2M}{2M\overline{D}} = 2 - \frac{1}{\overline{D}}$ and the result is proven. In case $m_a > m_d$, we simply switch the roles of m_a and m_d in the definition of δ and the numbers s_d. The computations are then exactly the same. □

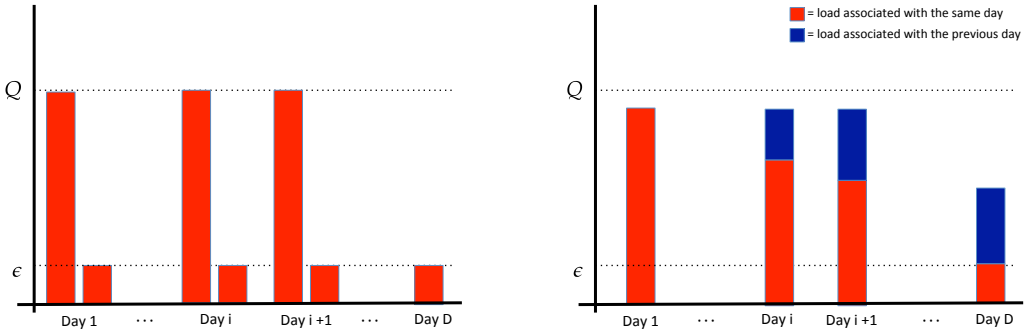


Figure 5.2: We illustrate the optimal solution with $\alpha = 0$, on the left, and with $\alpha = 1$, on the right. Each vertical bar represents the load of a vehicle. The red part of the bar is relative to the load associated with the fraction quantity q_d delivered on day d, i.e., $s_d q_d$. The blue part is the delivery associated with the day before, i.e., $(1 - s_{d-1})q_{d-1}$. Note that, when splitting is allowed, whether the load equals Q or not depends on the particular values of m_a, m_d and δ.

Property 5 says that, under very mild assumptions, the savings due to introducing the possibility of splitting the deliveries over consecutive days, in addition to splitting among vehicles only, can be up to 50%. Moreover, this result holds regardless of the value $m_a > 0$.

5.6 Mathematical Formulation and Solution Method

We now present a mathematical programming formulation for the MDSDVRP together with a few valid inequalities and a simple heuristic to construct a start solution. For modelling purposes we denote by $V = \{1, \ldots, \overline{V}\}$ the set of available vehicles. Since we assume unlimited available vehicles, we set $\overline{V} = \max_d |C_d|$. Note, however, that in consideration of Property 3, we could consider a different, and possibly lower, $\overline{V}$ without losing generality.

We make use of the flow variables

$$x_{ijv}^{d} = \begin{cases} 1, & \text{if vehicle } v \text{ travels from } i \text{ to } j \text{ on day } d \\ 0, & \text{otherwise} \end{cases}$$

Similarly, we denote by l_{ijv}^{d} the load of vehicle v on the arc (i,j) on day d. Let F be an integer variable representing the number of vehicles in the fleet. Finally, we introduce the continuous variables y_{iv}^{s} representing the fraction of demand q_i delivered by vehicle v on day $d_i + s\,(= d_i^s)$. To keep the notation simple, we include variables associated with self-pointing arcs, i.e. x_{iiv}^{d}, in summations. However, we do not include them in the implementation of the model. The model follows:

(**M**) minimise $$\sum_{d,i,j,v} x_{ijv}^{d} c_{ij} + Fb \tag{5.2}$$

subject to
$$\sum_{j \in \overline{P}_d} x_{ijv}^{d} = \sum_{j \in \overline{P}_d} x_{jiv}^{d} \qquad \forall\, i \in C,\, d \in D(i),\, v \in V \tag{5.3}$$

$$\sum_{j \in \overline{P}_{d_i^s}} x_{ijv}^{d_i^s} \geq y_{iv}^{s} \qquad \forall\, i \in C,\, s \in A,\, v \in V \tag{5.4}$$

$$m_a \sum_{j \in \overline{P}_{d_i^s}} x_{ijv}^{d_i^s} \leq y_{iv}^{s} \qquad \forall\, i \in C,\, s \in A,\, v \in V \tag{5.5}$$

$$\sum_{v \in V} y_{iv}^{0} \geq m_d \qquad \forall\, i \in C \tag{5.6}$$

$$\sum_{v \in V} \sum_{s \in A} y_{iv}^{s} = 1 \qquad \forall\, i \in C \tag{5.7}$$

$$\sum_{j \in \overline{P}_d} x_{ijv}^{d} \leq 1 \qquad \forall\, i \in C,\, v \in V,\, d \in D(i) \tag{5.8}$$

$$\sum_{s \in A} \sum_{i \in C_{d-s}} q_i\, y_{iv}^{s} \leq Q \qquad \forall\, v \in V,\, d \in D \tag{5.9}$$

$$l_{ijv}^{d} \leq Q\, x_{ijv}^{d} \qquad \forall\, i,j \in C,\, d \in D(i),\, v \in V \tag{5.10}$$

$$\sum_{v \in V} \sum_{j \in P_d} x_{0jv}^{d} \leq F \qquad \forall\, d \in D \tag{5.11}$$

$$\sum_{j \in \overline{P}_{d_i^s}} (l_{jiv}^{d_i^s} - l_{ijv}^{d_i^s}) = q_i\, y_{iv}^{s} \qquad \forall\, i \in C,\, s \in A,\, v \in V \tag{5.12}$$

$$\sum_{j \in P_d} l_{0jv}^{d} = \sum_{s \in A} \sum_{i \in C_{d-s}} q_i\, y_{iv}^{s} \qquad \forall\, d \in D,\, v \in V \tag{5.13}$$

$$F \in [0, \overline{V}], integer \tag{5.14}$$

$$x_{ijv}^{d} \in \{0,1\} \qquad \forall\, i,j \in C,\, v \in V,\, d \in D \tag{5.15}$$

$$l^d_{ijv} \in [0, Q] \qquad \forall\, i, j \in C,\ v \in V,\ d \in D \qquad (5.16)$$

$$y^s_{iv} \in [0, 1] \qquad \forall\, i \in C,\ v \in V,\ s \in A \qquad (5.17)$$

The objective function (5.2) is the sum of two terms, representing, respectively, the routing and fixed cost. Constraints (5.3) are the standard flow conservation constraints. The x and y variables are linked by constraints (5.4). Constraints (5.5) and (5.6) are minimum amount delivery constraints, respectively, for single deliveries and delivery on the first day. Constraints (5.7) require that the total demand must be delivered. Constraints (5.8) limit the visits per day per customer of each vehicle to one. Capacity constraints are expressed in (5.9) and (5.10). Constraints (5.11) link the daily fleets to the F variable. The load and y variables are linked in constraints (5.12) and (5.13). Finally, constraints (5.14 - 5.17) impose bounds and integrality on all variables.

5.6.1 Indivisible goods

As previously mentioned, we are allowing a delivery to be split fractionally. Consider the case of delivered goods being indivisible units. It is reasonable to assume the demands q_i and capacity Q are integers. Constraints (5.5) and (5.6) can be strengthen to account for a minimum number of units. For example, constraints (5.6) can be replaced by

$$\sum_{v \in V} y^0_{iv} \geq \frac{\lceil m_d q_i \rceil}{q_i} \qquad (5.6')$$

We denote by $\mathbf{M}'$ the model obtained by strengthening constraints (5.5) and (5.6). We can model indivisible units by replacing constraints (5.17) with

$$y^s_{iv} \in \{0, 1/q_i, \ldots, 1\} \quad \forall\, i \in C,\ v \in V,\ s \in A \qquad (5.17')$$

By applying a similar argument to the one presented in Archetti et al. [2006b] to prove Theorem 1 therein, it is possible to prove the following result

Theorem 1. *If $m_d = 0$, and problem $\mathbf{M}'$ has feasible solutions, then there exists an optimal solution satisfying constraints (5.17′).*

Proof. Let $\overset{*}{s}$ denote an optimal solution of $\mathbf{M}'$, let $\overset{*}{z}$ be its value, and let $\overset{*}{x}$ and $\overset{*}{y}$ be the corresponding optimal values of the variables. If all variables $\overset{*}{y}$ satisfy constraints 5.17′, then there is nothing to prove. Otherwise, let Y denote the set $\{(i, v, d) \subseteq C \times V \times D \mid \sum_{j \in \overline{P}_d} x^d_{ijv} \geq 1\}$. For each customer i and pair (v, d) define the sets $N(i) = \{(v, d) \mid (i, v, d) \in Y\}$ and $N(v, d) = \{i \mid (i, v, d) \in Y\}$. Let us introduce new variables $\widetilde{y}^s_{iv} = q_i y^s_{iv} - \lfloor q_i \overset{*}{y}^s_{iv} \rfloor$. Intuitively, we use these variables to reassign a part

of the deliveries, hence modifying the load of the vehicles, while keeping fixed the integer part of the deliveries of $\overset{*}{s}$. This is described by the following set of constraints

$$\sum_{i \in N(v,d)} \widetilde{y}^s_{iv} \leq Q - \sum_{i \in N(v,d)} \lfloor q_i \overset{*}{y}^s_{iv} \rfloor \qquad \forall\, v \in V, d \in D \tag{5.18}$$

$$\sum_{(v,d) \in N(i)} \widetilde{y}^s_{iv} = q_i - \sum_{(v,d) \in N(i)} \lfloor q_i \overset{*}{y}^s_{iv} \rfloor \qquad \forall\, i \in C \tag{5.19}$$

$$\widetilde{y}^s_{iv} \geq 0 \qquad \forall\, (i,v,d) \in Y \tag{5.20}$$

Constraints (5.18) - (5.20) are the classical ones of an assignment problem. A feasible assignment is defined by the fractional parts of $q_i\, \overset{*}{y}^s_{iv}$. Since the right-hand side is integer, it is well-known that there exists a feasible solution such that each variable $\widetilde{y}^s_{iv}$ has an integer value. If we replace the y^s_{iv} variables in $\overset{*}{s}$ by $(\widetilde{y}^s_{iv} + \lfloor q_i \overset{*}{y}^s_{iv} \rfloor)/q_i$ we have a new solution s' with the same routes of $\overset{*}{s}$. Each delivery in s' is integer and bigger or equal to the integer part of the associated delivery in $\overset{*}{s}$. Consequently, s' satisfies constraints (5.5′) and constraints (5.17′). Finally, the cost of s' is still $\overset{*}{z}$, since variables y^s_{iv} do not appear in the objective function. □

5.6.2 Valid Cuts

In what follows we present a set of valid inequalities derived from the literature and adapted to our formulation. The last four classes of cuts aim to break the symmetry of the vehicles.

Fractional Cycle Elimination These cuts, adapted from Dror et al. [1994], impose that if a customer is visited by a vehicle, then the vehicle has to leave the customer.

$$x^d_{ijv} \leq \sum_{p \in \overline{P}_d \setminus \{i\}} x^d_{jpv}, \quad \forall\, d \in D,\, i,j \in P_d$$

Minimum Number of Visits These cuts, adapted from Dror et al. [1994], provide a lower bound on the minimum number of times a customer has to be visited on the day of request.

$$\sum_{j \in \overline{P}_{t_i}, v \in V} x^{d_i}_{ijv} \geq \lceil q_i\, m_d \rceil \quad \forall\, i \in C$$

Depot Outgoing Degree These cuts are adapted from Dror et al. [1994]. Suppose for each day d we have a lower bound $\underline{v}_d$ on the number of vehicles that must be used on that day. The following inequalities state that the first $\underline{v}_d$ vehicles have to

leave the depot on day d:

$$\sum_{j \in P_d} x^d_{0jv} = 1 \quad \forall\, d \in D,\, v = 1, \ldots, \underline{v}_d.$$

Additionally, we impose that the following vehicles leave only if they deliver something. We denote with $\overline{v}_d$ an upper bound on the number of vehicle for day d:

$$\sum_{j \in P_d} x^d_{0jv} \geq \sum_{i \in P_d} y^d_{iv} \quad \forall\, d \in D,\, v = \underline{v}_d, \ldots, \overline{v}_d$$

Vehicle Usage This symmetry-breaking cut, inspired by Maheo et al. [2016], states that if a vehicle v is not used on day d, then all following vehicles cannot be used:

$$\sum_{i \in P_d} x^d_{0iv} \leq \sum_{i \in P_d} x^d_{0i(v-1)} \quad \forall\, d \in D,\, v = \underline{v}_d, \ldots, \overline{v}_d$$

Variable Fixing Following the idea presented in Dror et al. [1994], for each day we assign to the first vehicle the customer $i^*_d \in C_d$ which is the furthest from the depot among those whose original date of request is d:

$$\sum_{j \in \overline{P}_d} x^d_{i^*_d j0} = 1 \quad \forall\, d \in D$$

Visits Order This cut is inspired by Maheo et al. [2016]. We impose that, on day d, a vehicle v can visit a customer $i \in P_d$ only if vehicle $v-1$ visits any customer in the set $[1, \ldots, i] \cap P_d$. In order to make these cuts consistent with the *Variable Fixing* ones we do not consider the first vehicle. For all $d \in D$, $i, j \in P_d$, $v = 2, \ldots, \overline{v}_d$:

$$x^d_{ijv} \leq \sum_{\substack{e \in [1,\ldots,i] \cap P_d \\ s \in \overline{P}_d}} x^d_{se(v-1)}$$

Edge Sense These cuts were proposed in Dror et al. [1994]. On each day we fix an arc (s_d, e_d) with $s_d, e_d \in C_d$ and we impose that this arc can be traversed only in one direction. We pick the arc compatible with capacity constraints that minimises traversing cost.

$$\sum_{v \in V} x^d_{e_d s_d v} = 0 \quad \forall\, d \in D$$

Note that the *Edge Sense* cuts are not valid if the distance matrix is not symmetric.

5.6.3 Algorithm Description

The size of the model increases very quickly. Indeed, for relatively small $\overline{D}$ and N, if customers order frequently we already have a challenging problem. Solving model **M**, even when the formulation is enriched with the presented valid inequalities, can become very slow and we struggle to solve bigger instances. Therefore, in order to speed-up the process, we propose a simple heuristic to construct an initial solution.

The heuristic is divided in three phases. We first transform the MDSDVRP instance in a CVRP with multiple time windows. The idea is to force the multi-day structure by having disjoint time windows and the split-delivery structure by breaking each customer into a number of sub-customers, each supplying a m_a proportion of the total demand. We set the travelling time equals to the cost of the arcs and assume each vehicle leaves the depot at time 0. We assume that $1/m_a$ and m_d/m_a are integer, although the same idea can be applied even if this does not hold. Each customer in $i \in C$ is split into $1/m_a$ sub-customers with demand $q_i m_a$. We define time windows for each sub-customer as follows: define a large enough width Δ, and denote by tw_d, for each $d \in D$, the time window $[\Delta(d-1), \Delta d]$. Fix an original customer $i \in C$. All the sub-customers originated by i are assigned time window tw_{d_i}. Moreover, the last $(1-m_d)/m_a$ sub-customers are assigned α additional time windows $tw_{d_i+s}, s = 1, \ldots, \alpha$. Finally, we allow every vehicle to perform $\overline{D}$ trips, one per time window.

The second phase consists in solving the so transformed problem by means of an Adaptive Large Neighbourhood Search (ALNS)-based algorithm[2]. We use the method described in Kilby and Verden [2011a]. However, even though that method is capable of more specialised visit selection and ordering heuristics, it was tuned in a manner that replicates the ALNS procedure of Ropke and Pisinger [2006]. We use the *Shaw, random,* and *worst* destroy heuristics and the *greedy* and 2-, 3- and 4-regret insertion heuristics as defined in Ropke and Pisinger [2006].

During the search process, we record every "new best" solution produced by the algorithm. That is, as the search finds a solution with lower objective than the previous best, it is recorded. Thus, when the algorithm ends, we have a pool of routes that we use to create a solution using a MIP model. The idea is to use the routes' structure, to re-assign the quantities delivered by each route to each visited customer and to choose the best set of routes. Hence the quantities delivered by the routes become decision variables. Let us denote by R the set of routes recorded, and by R_d the subset of routes on day d. For each route $r \in R$ we denote by c_r its routing cost. Moreover, for each customer $i \in C$, we introduce a binary parameter δ_{ir} that indicates

[2]For a presentation of ALNS see Section 3.4.1.

whether r visits i, a binary variable u_r, representing whether or not the route is used, and an integer variable y_{ir}, quantifying how many minimum portions $m_a q_i$, does the route r deliver to i, i.e., the amount delivered is $y_{ir} m_a q_i$. The model follows:

$$(\mathbf{RM}) \text{ minimise } \sum_{r \in R} c_r u_r + Fb \tag{5.21}$$

$$\text{subject to } \sum_{r \in R_d} u_r \leq F \qquad \forall\, d \in D \tag{5.22}$$

$$\sum_{r \in R} y_{ir}\, \delta_{ir} = \frac{1}{m_a} \qquad \forall\, i \in C \tag{5.23}$$

$$\sum_{r \in R_d} y_{ir}\, \delta_{ir} \geq \frac{m_d}{m_a} \qquad \forall\, i \in C \tag{5.24}$$

$$y_{ir} \leq \frac{u_r}{m_a} \qquad \forall\, i \in C, r \in R \tag{5.25}$$

$$\sum_{i \in C} y_{ir}\, m_a\, q_i \leq Q \qquad \forall\, r \in R \tag{5.26}$$

$$F \in [0, \overline{V}], integer \tag{5.27}$$

$$y_{ir} \in [0, \frac{1}{m_a}], integer \qquad \forall\, i \in C, r \in R \tag{5.28}$$

$$u_r \in \{0, 1\} \qquad \forall\, r \in R \tag{5.29}$$

The objective (5.21) is the sum of routing and fixed costs. Constraints (5.22) force that the overall fleet is big enough to cover the routes on all days. Constraints (5.23) and (5.24) impose that every request is satisfied and a fixed minimum percentage is delivered on the day of request. Constraints (5.25) link the u and y variables. Constraints (5.26) ensure the capacity is not exceeded. Integrality and bounds on the variables are imposed in constraints (5.27) - (5.28). Note that since y_{ir} can assume the value zero, model RM can essentially "skip" a customer in the route.

A similar idea was proposed in Archetti et al. [2008] where the authors use the information provided by a tabu search to locate promising area in the search space. These areas are then better explored through the use of an integer programming model. Our heuristic is based on the same idea even though at a very basic level. An advanced version could use the arcs in the routes set as an input to the **M** model. However, the resulting model would still be challenging even for rather small instances.

In summary, the method we use to solve an instance of the MDSDVRP is described in Algorithm 5. We use a ALNS-based algorithm to localise a promising area of the search space, i.e., to generate a pool of routes. Subsequently, we solve model **RM** to further explore the identified area. The solution so obtained is used to warm

start the enriched version of model **M**.

Algorithm 5

Input: An instance of MDSDVRP

1: transform the problem into a CVRP with multiple TW
2: run ALNS-based algorithm (Kilby and Verden [2011a]) and record all "new-best" solutions found
3: solve model **RM** using the routes found as columns
4: initialise **M** with the solution of **RM**
5: solve model **M** enriched with the cuts presented in Section 5.6.

5.7 Computational Analysis

In this section, we aim at presenting an experimental analysis of the strategy proposed. The first goal is to illustrate the possible benefits. We also intend to analyse which features have the greatest impact of the effectiveness of the approach. This is fundamental to understand when this strategy can be applied with substantial savings. We will analyse both the cases $b = 0$ and $b > 0$. The former considers an unlimited fleet with no fixed costs, hence it focuses only on the routing part of the problem. The latter includes a fleet size component. We think that an analysis of the approach has to be done also considering the pure routing problem, especially because in the literature this type of analysis has been always done (Archetti et al. [2006a, 2015b]; Dror et al. [1994]; Gulczynski et al. [2010]) focusing on the former case. We will refer to the two cases as *routing* and *fleet size* case.

5.7.1 Instances

There are various sets of benchmarks for multi-day problems, however, they are tailored mainly for the IRP or PVRP. We decided to create some new instances to study how beneficial splitting the demands over days can be in practice. In our instances, the number N of original customers (which may place requests on different days) ranges in the set $\{2, 3, 5, 7, 10, 15\}$ and the number of days in $\{3, 5, 7\}$. We now present the main features of the instances.

- **Locations.** We have two types of instances. In the first type, the locations are chosen randomly in a square of side 80 with the depot in the centre. In the second type the locations form two different clusters. We refer to the customers' layout as *map* and write *random* and *cluster* to respectively refer to the two set of locations.

- **Demand.** The customers are randomly divided in two categories: "big" and "small". Each customer has a "reference" demand which is randomly chosen in one of the intervals $[5, 15], [25, 50]$, respectively, if the customer is big or small. Whenever a customer places an order on the day the request is randomly chosen using a Gaussian distribution $\mathcal{N}(q, 2)$ where q is the reference demand.
- **Frequency.** The number of requests per customers varies from 1 to $\overline{D}$. Hence the minimum number of total possible requests is N while the maximum is $n\overline{D}$. We introduce a parameter $p \in [0, 1]$. For a given p we randomly choose $\lfloor pN(\overline{D} - 1) \rfloor + N$ of the possible requests with the guarantee that each customer has at least one request. We created instances with $p \in \{0.3, 0.7\}$
- **Other parameters.** We set the capacity $Q = 40$, splitting horizon $\alpha = 1$, and minimum delivery amount parameters: $m_a = 0.1, m_d = 0.5$. In the fleet size case we set a fixed cost per day $b_d = 70$, therefore we have $b = \overline{D} b_d$

Later, in in the section we will provide an analysis of the effect of different features (clustering, frequency, m_d, splitting horizon) on the total cost.

For each scenario (fixed tuple of parameters) we create 5 instances[3] . We solved each instance by applying Algorithm 5. The ANLS-based algorithm described in Kilby and Verden [2011a] was coded in a C++ environment. We used the Python API of Gurobi 6.5 (Gurobi Optimization [2015]) to solve the models RM and M. We stopped the execution after 4 hours. In all the following experiments, we compare the case $\alpha > 0$ to the no-split case ($\alpha = 0$) to analyse the effect of allowing splits over days. In particular, we look at the percentage savings, with respect to the no-split case, in the routing cost (Δ-Rout) and the total cost (Δ-Tot), and the difference in the fleets (Δ-Fleet) between the two cases. Obviously, in the routing case, the fleet does not play any role in the final cost and savings. In Table 5.1 and 5.2 we summarise the results for, respectively, the routing case and fleet size case. The parameters describing the instance are given in the first four columns. We recall that $|C|$ represents the total number of requests in the instance (which throughout the chapter we have identified with customers), while N is the number of original customers that place requests on several days. The next 5 columns represent the cluster instances, the last 5 the random instances. For each scenario we report the average, over the 5 instances, of the optimality gap (of model **M**) for the case with no split over days ($\alpha = 0$) and with split over days allowed ($\alpha = 1$), the savings in routing (Δ-Rout) and total (Δ-Tot) cost, and the difference of fleet (Δ-Fleet).

We first observe that the optimality gap reached by the model is always low in the case $\alpha = 0$. If we allow split deliveries over days the gap increases substantially

[3]The instances are available online at https://fbertoli.github.io/downloads

for the biggest instances ($N = 10, 15$). It is clear that, since setting $\alpha = 1$ yields a relaxation of the case $\alpha = 0$, the total savings can only be positive. This is not always the case in practice, and it is due to the fact that the optimality gap can be substantially higher in the former case. A few remarks are in order: the savings appear to be lower if the locations are clustered. The instances become more challenging for $p = 0.7$ and this could hide some possible gain of the strategy. Overall, the results are encouraging. The savings resulting from allowing deliveries split over days can be substantial. It is clear that introducing a fleet size component in the problem can create more possibilities for savings. This is due to the fact that the fixed cost is comparable with the routing cost. However, the results suggest that even in case where the fixed cost is zero the strategy can be effective.

In next sub-sections, we examine the influence on the total savings of the customers' location, the frequency of ordering, the minimum amount constraints and of the length of the splitting horizon. We decided to consider only instances with $N = 10$. The reason for this choice is that these instances are amongst the biggest, but we still can reach low optimality gap. Hence, they can provide better insights. In each sub-section, we will vary some of the parameters. If not stated otherwise, the remaining parameters are the ones described above.

5.7.2 Minimum delivery amount constraints

In this sub-section, we aim at studying the impact of the minimum delivery amount constraint on the day of request. We solve each instance with different values of m_d. In Tables 5.3 and 5.4, respectively for the routing and fleet size case, we report, for each instance and value of m_d, the average, out of 5 instances, of the optimality gap for the case $\alpha = 1$ and the savings in the total cost (Δ-Tot) with respect to the case $\alpha = 0$, both expressed in percentage. We do not report the optimality gaps for the case $\alpha = 0$ as these are all negligible. If the linear relaxation of $\mathbf{M}$ could not be solved within the cut-off time we do not consider it in the average.

Again, lowering m_d can only result in higher savings, but this is not always observed in the results, due to the fact that instances with low m_d and high $\overline{D}$ are computationally more challenging. However, the expected behaviour of higher savings corresponding to lower values of m_d is clear in the tables. In particular, it is very pronounced in the fleet size case. In this case, lowering m_d can triple the savings (Δ) with respect to the routing case.

5.7.3 Frequency

We now focus on the frequency of service. In order to do so we vary the frequency parameter p in the set $\{0.1, 0.2, \ldots, 1\}$. In Table 5.5 we report the results. Similarly to the previous sub-sections we report only the average optimality gap for the case $\alpha = 1$ and the average total savings for each instance.

We observe that the highest savings correspond to low-medium values of p. It has to be noted that for high value of p the optimality gap is significantly higher and could disturb the interpretation of the data. Although it seems intuitive that a high value of p does not allow very high savings. This could be explained as follows: a customer that orders every day will have to be visited every day. Therefore, splitting the demand over days can have the effect of smoothing the peaks and reduce the original number of splits, but it will not change the routes substantially.

5.7.4 Clustering

In this sub-section, we focus on the customers' layout. In order to focus on the layout only we proceed as follows: we define a smooth transformation, parametrised by $\theta \in [0, 1]$. This changes the cluster map into a randomised map. Then we solve the instances, with $p \in \{0.3, 0.7\}$, using the maps defined by $\theta \in \{0, 0.1, 0.2, \ldots, 1\}$. This gives us a better idea of what happens when the customers are moved from a clustered to a completely random structure.

The transformation is described next. Consider the cluster map. For the two clusters identify a square, of minimum size $\sigma_i, i = 1, 2$, centred on the cluster's centre δ_i, that contains all customers belonging to that cluster. Note that the two squares could, and in fact do, overlap. For any $\theta \in [0, 1]$ we do the following:

- move the clusters' centres on a line joining them with the depot linearly with θ;
- for each customer, consider the vector $\overline{v}$ joining it with the cluster's centre δ_i, and relocate the customer in $\delta_i + (\theta(80 - \sigma_i)/\sigma_i + 1)\overline{v}$. In other words, each customer is translated to maintain its angle with the cluster's centre δ_i, while the distance from δ_i is progressively expanded, or shrunk.

If $\theta = 0$ we do not alter the cluster map. As θ increases to 1 the cluster's centres collide with the depot and the squares containing the customer expand (or shrink) to a square of side 80, which is the same length as the square's side we used for the random map instances. In Table 5.6 we report the results for different values of θ with the same convention we used in previous tables. The trend is clear: the savings are higher if the customers are clustered. However, the decreasing rate is not as drastic as one could expect, the difference between the two extremes vary between

$1-5$ points in percentage. One other observation is that the optimality gap increases with θ, thus indicating that a non-clustered map is harder to solve.

5.7.5 Longer Splitting Horizon

Lastly, we consider longer splitting horizons by letting α vary in the set $\{0, 1, 2, 3\}$. The results are reported in Table 5.7 to 5.10. Note that setting $\alpha = 3$ is not applicable (n.a.) if $\overline{D} = 3$. Increasing the splitting horizon can only lead to higher savings from a theoretical point of view but it also increases the computational complexity dramatically. However, it can be seen that bigger values of α can lead to substantial gains.

We note that most of the benefit comes from setting $\alpha = 1$. That is, a small change in the operational practice can have a large benefit on the costs, while the problem remains computationally tractable. We believe this validates the approach, as splitting a delivery over two consecutive days will very often not cause problems for customers.

5.8 Contributions and Conclusion

In this chapter, we proposed a new problem, the MDSDVRP. The problem captures the operations of a transportation agent that delivers products to many customers from a single distribution centre. The planning horizon is made of several days and the customers involved may place orders on more than one day. The decisions to be made concern two competing aspects of the problem: the fleet size and the routing schedule.

We provided a theoretical analysis of the problem. We considered some known properties of optimal solutions of an SDVRP and show that some can be extended while others fail to hold when splits over days are allowed.

We proposed a mathematical programming formulation for the MDSDVRP together with a simple heuristic to warm start the optimisation process. We also proposed adaptation of existing valid cuts to strengthen the formulation.

We proposed a series of experiments aimed at investigating the possible benefits of the proposed approach, and also at quantifying the influence that different features of an instance (such as degree of clustering, frequency of service and minimum amount constraints) have on the efficiency of the strategy.

We observed that by splitting the deliveries over consecutive days, substantial savings can be achieved, both in terms of the routing and the fixed costs.

We noted that this strategy could be applied in practise, even when an integration

between the inventory management and the routing component of the problem at hand is difficult or impractical. In support of this hypothesis, we noted that most of the benefit comes from allowing split deliveries over only 2 consecutive days. To the best of our knowledge, no paper has theoretically analysed the advantages of a similar approach, and, with the partial exception of Archetti et al. [2015b], no paper has proposed a computational analysis. Our study is based on and inspired by real problems.

Future research could aim at strengthening Property 5 (i.e., removing the additional assumption and finding a tight bound) and extending the approach to real-world scenarios. This would require to consider bigger instances and a fleet composition aspects, rather than only focusing on the size of the fleet.

Instance				cluster					random				
$\overline{D}$	N	p	$\|C\|$	$\alpha = 0$ gap (%)	$\alpha = 1$ gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	$\alpha = 0$ gap (%)	$\alpha = 1$ gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)
2	3	0.3	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	5	0.3	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	7	0.3	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	3	0.3	4	0.00	0.00	11.74	0.40	11.74	0.00	0.00	4.95	0.60	4.95
3	5	0.3	6	0.00	0.00	16.52	0.40	16.52	0.00	0.00	12.06	0.40	12.06
3	7	0.3	7	0.00	0.00	16.00	0.60	16.00	0.00	0.00	6.84	0.20	6.84
5	3	0.3	8	0.00	0.00	12.78	0.75	12.78	0.00	0.00	7.90	0.40	7.90
5	5	0.3	11	0.00	0.00	19.15	0.60	19.15	0.00	0.00	4.44	0.00	4.44
5	7	0.3	13	0.00	0.00	15.77	0.60	15.77	0.00	0.00	5.58	0.40	5.58
7	3	0.3	12	0.00	0.00	16.21	0.20	16.21	0.00	0.00	7.28	0.40	7.28
7	5	0.3	16	0.00	0.00	13.64	0.40	13.64	0.00	0.00	8.84	0.40	8.84
7	7	0.3	19	0.00	0.00	14.29	0.80	14.29	0.00	0.00	7.29	0.00	7.29
10	3	0.3	18	0.00	0.05	7.70	0.40	7.70	0.00	0.02	7.74	0.00	7.74
10	5	0.3	23	0.00	0.02	13.84	0.60	13.84	0.00	0.00	5.44	0.20	5.44
10	7	0.3	28	0.00	0.02	15.31	0.80	15.31	0.00	0.00	6.31	0.60	6.31
15	3	0.3	27	0.00	0.05	7.98	0.00	7.98	0.00	0.12	4.02	0.40	4.02
15	5	0.3	36	0.00	0.04	11.01	0.80	11.01	0.00	0.07	5.94	0.20	5.94
15	7	0.3	44	0.00	0.04	11.78	0.80	11.78	0.00	0.06	6.01	0.20	6.01
2	3	0.7	4	0.00	0.00	0	0	0	0.00	0.00	0.00	0.00	0.00
2	5	0.7	5	0.00	0.00	3.56	0.20	3.56	0.00	0.00	1.30	0.20	1.30
2	7	0.7	6	0.00	0.00	2.21	0.25	2.21	0.00	0.00	2.85	0.40	2.85
3	3	0.7	7	0.00	0.00	13.24	1.00	13.24	0.00	0.00	10.25	0.60	10.25
3	5	0.7	10	0.00	0.00	15.89	0.80	15.89	0.00	0.00	13.77	1.00	13.77
3	7	0.7	12	0.00	0.00	19.32	0.80	19.32	0.00	0.00	15.75	0.40	15.75
5	3	0.7	13	0.00	0.01	11.62	0.00	11.62	0.00	0.00	8.05	0.20	8.05
5	5	0.7	19	0.00	0.01	15.46	0.00	15.46	0.00	0.01	12.85	0.00	12.85
5	7	0.7	24	0.00	0.02	15.98	0.20	15.98	0.00	0.03	12.95	0.00	12.95
7	3	0.7	19	0.00	0.06	9.31	0.40	9.31	0.00	0.04	12.93	0.20	12.93
7	5	0.7	28	0.00	0.06	12.54	0.40	12.54	0.00	0.04	12.09	0.40	12.09
7	7	0.7	36	0.00	0.06	12.31	0.20	12.31	0.00	0.05	10.75	-0.20	10.75
10	3	0.7	18	0.01	0.08	10.53	0.40	24.90	0.00	0.10	9.68	0.00	9.68
10	5	0.7	41	0.00	0.09	7.29	0.20	7.29	0.00	0.18	8.53	0.00	8.53
10	7	0.7	54	0.00	0.10	11.05	-0.20	11.05	0.00	0.21	7.47	0.00	7.47
15	3	0.7	44	0.02	0.13	1.96	-0.40	1.96	0.03	0.24	0.09	-0.20	0.09
15	5	0.7	64	0.02	0.13	1.16	0.00	1.16	0.06	0.28	-3.80	0.00	-3.80
15	7	0.7	84	0.02	0.17	-0.66	-0.60	-0.66	0.04	0.30	-3.37	-0.20	-3.37
					Average =	10.5	0.32	10.5			6.46	0.2	6.46

Table 5.1: Routing case ($b_d = 0$). Savings with respect to the "no-split" case.

Instance				cluster					random				
h	n	p	|C|	$\alpha = 0$ gap (%)	$\alpha = 1$ gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	$\alpha = 0$ gap (%)	$\alpha = 1$ gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)
2	3	0.3	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	5	0.3	3	0.00	0.00	-0.61	0.20	9.54	0.00	0.00	0.00	0.00	0.00
2	7	0.3	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	3	0.3	4	0.00	0.00	11.74	0.40	14.69	0.00	0.00	4.83	0.80	21.12
3	5	0.3	6	0.00	0.00	16.52	0.40	17.52	0.00	0.00	9.77	0.60	21.26
3	7	0.3	7	0.00	0.00	16.00	0.60	20.53	0.00	0.00	5.56	0.80	20.53
5	3	0.3	8	0.00	0.00	16.47	0.80	23.21	0.00	0.00	6.39	0.80	17.53
5	5	0.3	11	0.00	0.00	19.01	0.80	25.70	0.00	0.00	3.01	0.40	10.16
5	7	0.3	13	0.00	0.00	15.72	0.80	23.57	0.00	0.00	5.58	0.40	11.64
7	3	0.3	12	0.00	0.00	15.06	0.80	22.21	0.00	0.00	4.90	0.80	16.65
7	5	0.3	16	0.00	0.00	13.64	0.40	15.29	0.00	0.00	7.18	0.80	17.98
7	7	0.3	19	0.00	0.00	14.29	0.80	22.80	0.00	0.00	6.41	0.40	11.81
10	3	0.3	18	0.00	0.01	7.17	0.60	13.27	0.00	0.00	5.77	0.80	14.19
10	5	0.3	23	0.00	0.00	13.77	0.80	21.60	0.00	0.00	4.19	0.80	15.58
10	7	0.3	28	0.00	0.00	15.12	1.00	24.35	0.00	0.00	5.30	1.00	20.58
15	3	0.3	27	0.00	0.01	7.64	0.80	13.53	0.00	0.03	5.87	1.20	15.41
15	5	0.3	36	0.00	0.02	10.60	1.00	18.31	0.00	0.01	6.59	1.20	17.95
15	7	0.3	44	0.00	0.02	10.34	1.60	24.39	0.00	0.03	5.96	0.80	13.67
2	3	0.7	4	0.00	0.00	-0.71	0.20	7.96	0.00	0.00	0.00	0.00	0.00
2	5	0.7	5	0.00	0.00	3.56	0.20	10.10	0.00	0.00	1.30	0.20	8.49
2	7	0.7	6	0.00	0.00	2.47	0.20	9.46	0.00	0.00	2.85	0.40	16.03
3	3	0.7	7	0.00	0.00	13.24	1.00	24.32	0.00	0.00	8.65	1.00	21.57
3	5	0.7	10	0.00	0.00	15.82	1.00	25.49	0.00	0.00	13.77	1.00	24.02
3	7	0.7	12	0.00	0.00	19.14	1.00	27.11	0.00	0.00	14.93	1.00	24.76
5	3	0.7	13	0.00	0.00	11.62	0.00	6.20	0.00	0.00	3.81	0.80	15.01
5	5	0.7	19	0.00	0.00	15.06	0.40	14.19	0.00	0.00	10.18	0.40	11.58
5	7	0.7	24	0.00	0.01	15.93	0.40	14.57	0.00	0.00	12.66	0.20	9.86
7	3	0.7	19	0.00	0.00	9.17	0.60	13.36	0.00	0.00	12.93	0.20	10.01
7	5	0.7	28	0.00	0.02	12.39	0.80	17.52	0.00	0.01	9.74	0.60	13.98
7	7	0.7	36	0.00	0.02	12.02	0.80	18.03	0.00	0.02	10.45	0.40	11.08
10	3	0.7	28	0.00	0.03	10.15	0.60	12.54	0.00	0.02	9.61	0.60	12.17
10	5	0.7	41	0.00	0.02	8.18	0.80	14.29	0.00	0.12	3.27	0.40	6.35
10	7	0.7	54	0.00	0.04	11.60	0.40	10.81	0.00	0.14	7.08	0.40	8.48
15	3	0.7	44	0.01	0.06	3.57	0.40	5.10	0.01	0.12	1.55	0.60	5.90
15	5	0.7	64	0.01	0.12	0.57	0.40	3.76	0.03	0.19	-3.78	0.40	1.21
15	7	0.7	83	0.01	0.13	-0.22	0.40	3.46	0.02	0.25	-8.52	0.40	-0.83
					Average =	10.70	0.59	15.24			5.49	0.57	12.38

Table 5.2: Fixed cost case ($b_d = 70$). Savings with respect to the "no-split" case.

		$p = 0.3$						$p = 0.7$					
		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$	
	m_d	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot
cluster map	0.1	0.06	8.93	0.03	15.26	0.03	16.86	0.10	8.38	0.23	-3.90	-	-
	0.2	0.06	8.46	0.03	14.96	0.03	16.69	0.13	5.11	0.18	-1.05	0.10	9.58
	0.3	0.05	8.13	0.03	14.41	0.02	16.45	0.08	10.29	0.11	5.33	0.14	20.26
	0.4	0.05	7.96	0.02	14.16	0.02	15.70	0.08	9.83	0.09	7.70	0.21	-1.77
	0.5	0.05	7.70	0.02	13.72	0.02	15.29	0.09	9.46	0.08	7.50	0.12	10.17
	0.6	0.04	7.17	0.02	12.30	0.02	13.70	0.08	9.92	0.08	7.32	0.11	10.46
	0.7	0.02	6.77	0.01	10.51	0.00	12.61	0.08	9.51	0.08	5.75	0.09	10.17
	0.8	0.01	6.64	0.00	7.57	0.00	9.69	0.07	9.05	0.05	6.11	0.11	8.67
	0.9	0.00	6.88	0.00	3.28	0.00	5.14	0.07	5.57	0.03	5.51	0.06	5.63
random map	0.1	0.02	9.71	0.01	6.34	0.00	7.90	0.17	8.22	0.26	6.29	0.21	12.75
	0.2	0.01	9.75	0.01	6.13	0.01	7.66	0.16	7.59	0.33	4.65	0.25	5.17
	0.3	0.02	8.79	0.01	5.97	0.01	7.25	0.13	8.56	0.27	2.75	0.22	9.61
	0.4	0.03	8.15	0.01	5.66	0.01	6.88	0.13	9.10	0.20	9.08	0.21	7.75
	0.5	0.02	7.63	0.00	5.44	0.00	6.31	0.11	9.05	0.19	8.27	0.20	6.83
	0.6	0.02	7.40	0.00	5.41	0.01	5.38	0.13	7.36	0.18	6.53	0.20	5.20
	0.7	0.03	6.21	0.00	4.25	0.00	4.91	0.15	5.81	0.12	4.55	0.18	5.83
	0.8	0.03	4.46	0.00	4.19	0.00	4.38	0.08	5.90	0.08	5.96	0.10	4.59
	0.9	0.00	1.61	0.00	0.25	0.00	2.07	0.06	0.84	0.05	1.68	0.08	0.80

Table 5.3: m_d analysis - routing case. All entries in the table are percentages.

		$p = 0.3$						$p = 0.7$					
		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$	
	m_d	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot
cluster map	0.1	0.01	14.1	0	22.31	0	28.68	0.03	12.97	0.03	14.16	0.03	4.83
	0.2	0.01	13.85	0	22.03	0	28.58	0.03	12.93	0.06	11.3	0.11	15.65
	0.3	0.01	13.65	0	21.96	0	28.42	0.03	12.98	0.02	14.29	0.03	12.77
	0.4	0.01	13.56	0	21.79	0	27.91	0.03	12.84	0.02	14.38	0.04	11.29
	0.5	0.01	13.27	0	21.6	0	24.35	0.03	12.61	0.03	14.18	0.04	11.03
	0.6	0.01	6.99	0	20.83	0	20.13	0.03	12.42	0.02	14.07	0.03	10.95
	0.7	0	9.79	0	16.31	0	19.42	0.03	12.17	0.03	12.55	0.04	10.43
	0.8	0	7.04	0	14.91	0	14.88	0.03	11.95	0.03	10.7	0.05	9.4
	0.9	0	9.33	0	1.54	0	12.93	0.03	5.35	0.01	2.17	0.03	2.84
random map	0.1	0	18.79	0	19.64	0	21.87	0.03	12.87	0.1	7.16	0.17	11.37
	0.2	0	18.54	0	19.49	0	21.59	0.03	12.61	0.12	6.68	0.12	17.08
	0.3	0	17.72	0	19.13	0	21.3	0.03	12.52	0.12	7.05	0.14	9.09
	0.4	0	16.91	0	18.75	0	21.05	0.03	12.45	0.06	9.39	0.1	12.84
	0.5	0	14.19	0	15.58	0	20.58	0.03	11.94	0.06	11.27	0.09	13.06
	0.6	0	12.9	0	14.84	0	20.03	0.03	11.42	0.05	11.04	0.08	11.41
	0.7	0.01	10.95	0	11.51	0	16.44	0.05	9.99	0.05	9.45	0.09	7.93
	0.8	0	9.13	0	5.29	0	14.38	0.04	6.91	0.06	2.82	0.05	6.96
	0.9	0	3.49	0	3.34	0	8.21	0.02	3.15	0.03	0.84	0.02	3.2

Table 5.4: m_d analysis - fleet size case. All entries in the table are percentages.

		cluster						random					
		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$	
	p	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot
routing case	0.1	0	13.73	0	12.24	0	9.79	0	5.92	0	2.8	0	4.01
	0.2	0.01	12.05	0	18.25	0	13.49	0	5.14	0	7.05	0	5.4
	0.3	0.03	11.28	0.02	15.61	0.03	12.34	0	4.51	0	7.64	0.01	5.58
	0.4	0.07	9.41	0.04	12.25	0.04	12.3	0.03	6.54	0.01	7.98	0.03	8.15
	0.5	0.06	12.28	0.06	11.11	0.06	12.07	0.06	9.22	0.05	8.42	0.05	7.76
	0.6	0.07	10.36	0.06	11.17	0.06	10.56	0.09	8.8	0.08	9.37	0.09	6.75
	0.7	0.09	9.06	0.1	10.5	0.1	9.79	0.11	8.52	0.16	7.79	0.19	6.15
	0.8	0.1	6.72	0.11	9.06	0.13	7	0.17	6.87	0.22	6.63	0.22	6.49
	0.9	0.12	4.53	0.14	4	0.18	-0.63	0.23	6.26	0.26	1.76	0.25	3.83
	1	0.13	1.36	0.16	-1.08	0.25	-11.92	0.25	3.13	0.28	1.46	0.39	-13.71
fleet size case	0.1	0	18.44	0	18.49	0	17.98	0	18.14	0	14.2	0	11.22
	0.2	0	19.75	0	23.85	0	25.21	0	17.98	0	17.94	0	14.24
	0.3	0	18.42	0	21.6	0	24.53	0	17.53	0	18.84	0	16.4
	0.4	0.01	16.47	0.01	18.64	0.01	20.39	0	14.53	0	10.23	0.01	15.93
	0.5	0.01	14.32	0.02	16.91	0.03	19.37	0	14.76	0.02	12.42	0.03	17.37
	0.6	0.02	10.61	0.02	18.63	0.03	18.46	0.01	14.71	0.04	14.81	0.04	16.05
	0.7	0.03	7.36	0.02	16.18	0.05	17.09	0.02	9.57	0.04	11.58	0.1	11.94
	0.8	0.03	4.4	0.03	7.83	0.1	9.05	0.09	4.56	0.09	7.57	0.14	6.81
	0.9	0.02	3.44	0.06	3.66	0.13	-2.82	0.07	4.76	0.12	3.92	0.16	-1.25
	1	0.05	1.11	0.08	0	0.19	-13.59	0.12	2.15	0.15	-0.27	0.25	-6.82

Table 5.5: Frequency analysis. All entries in the table are percentages.

		$p = 0.3$						$p = 0.7$					
		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$		$\overline{D} = 3$		$\overline{D} = 5$		$\overline{D} = 7$	
	theta	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot	gap	Δ-Tot
routing case	0	0.05	7.7	0.02	13.84	0.02	15.22	0.09	9.39	0.1	6.98	0.11	10.73
	0.1	0.04	7.55	0.01	13.57	0.02	14.91	0.08	9.24	0.1	6.88	0.11	10.41
	0.2	0.04	6.9	0.01	11.99	0.02	14.16	0.09	8.69	0.11	5.85	0.12	9.79
	0.3	0.04	7.1	0.01	11.1	0.02	13.05	0.09	7.93	0.12	6.25	0.12	9.1
	0.4	0.02	6.94	0	10.56	0.02	11.91	0.09	7.53	0.13	6.7	0.14	7.81
	0.5	0.02	6.88	0	9.58	0.01	10.57	0.13	6	0.14	6.67	0.15	7.65
	0.6	0.01	7.18	0.01	8.16	0.01	9.52	0.09	7.55	0.15	7.42	0.17	7.53
	0.7	0.01	7.21	0	7.69	0.01	8.65	0.12	7.04	0.16	5.88	0.18	7.19
	0.8	0	7.03	0	6.99	0.01	8.26	0.13	7.38	0.16	5.73	0.18	8.48
	0.9	0	6.63	0	6.49	0.01	7.72	0.14	7.04	0.17	6.58	0.18	8.2
	1	0.01	6.88	0	6.11	0.01	7.87	0.1	7.72	0.19	5.2	0.2	8.47
fleet size case	0	0	13.27	0	21.6	0	24.35	0.03	12.61	0.03	14.21	0.03	11.05
	0.1	0	13.52	0	21.79	0	24.53	0.02	12.35	0.03	14.38	0.04	10.74
	0.2	0	13.43	0	21.21	0	24.53	0.02	12.2	0.03	14.17	0.04	10.15
	0.3	0	13.55	0	21.13	0	24.44	0.02	11.99	0.03	14.71	0.05	9.7
	0.4	0	13.45	0	20.79	0	24.06	0.02	11.88	0.03	14.92	0.05	9.46
	0.5	0	13.11	0	20.17	0	23.46	0.02	11.77	0.03	14.95	0.05	9.42
	0.6	0	12.83	0	19.37	0	22.77	0.04	10.59	0.05	14.45	0.06	9.57
	0.7	0	12.42	0	18.76	0	22.09	0.02	11.49	0.04	14.32	0.09	8.88
	0.8	0	12.01	0	17.99	0	21.41	0.02	11.37	0.07	13.45	0.06	9.17
	0.9	0	11.68	0	17.42	0	20.73	0.02	11.35	0.06	13.44	0.08	8.74
	1	0	11.58	0	16.65	0	20.1	0.02	11.5	0.05	12.87	0.09	9.14

Table 5.6: Customers' disposition analysis. All entries in the table are percentages.

Instance			$\alpha = 1$				$\alpha = 2$				$\alpha = 3$			
p	$\overline{D}$	m_d	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)
0.3	3	0.3	0.01	7.96	0.6	13.65	0.02	9.32	0.6	14.29	n.a.	n.a.	n.a.	n.a.
0.3	3	0.7	0	6.77	0.4	9.79	0	7.66	0.4	10.21	n.a.	n.a.	n.a.	n.a.
0.3	5	0.3	0	14.53	0.8	21.96	0.01	15.7	0.8	22.51	0.02	16.29	0.8	22.79
0.3	5	0.7	0	10.56	0.6	16.31	0	13.01	0.6	17.47	0.01	13.4	0.6	17.65
0.3	7	0.3	0	16.28	1.2	28.42	0.01	18.48	1.2	29.36	0.02	19.11	1.2	29.64
0.3	7	0.7	0	11.96	0.8	19.42	0	14.98	1	24.29	0.01	16.81	1	25.07
0.7	3	0.3	0.03	11.03	0.6	12.98	0.26	-11.8	-2	-30.58	n.a.	n.a.	n.a.	n.a.
0.7	3	0.7	0.03	9.44	0.6	12.17	0.07	8.59	0.6	11.74	n.a.	n.a.	n.a.	n.a.
0.7	5	0.3	0.02	8.18	0.8	14.29	0.17	0.56	0.5	6.73	0.1	6.05	0.67	11.54
0.7	5	0.7	0.03	5.93	0.75	12.55	0.07	6.05	0.8	13.26	0.13	3.06	0.6	9.23
0.7	7	0.3	0.03	13.03	0.5	12.77	0.26	-4.51	-1.5	-20.78	0.17	10.95	0	5.55
0.7	7	0.7	0.04	10.85	0.4	10.43	0.08	6.85	0.25	6.56	0.1	8.19	0.4	9.08

Table 5.7: Routing case - cluster map

Instance			$\alpha = 1$				$\alpha = 2$				$\alpha = 3$			
p	$\overline{D}$	m_d	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)
0.3	3	0.3	0	7.18	1	17.72	0.01	10.26	1	19.23	n.a.	n.a.	n.a.	n.a.
0.3	3	0.7	0.01	-0.86	0.8	10.95	0.02	5.99	0.8	14.3	n.a.	n.a.	n.a.	n.a.
0.3	5	0.3	0	4.52	1	19.13	0	9.55	1	21.61	0.05	10.14	1	21.9
0.3	5	0.7	0	2.78	0.6	11.51	0.01	6.38	0.6	13.28	0.03	7.44	0.6	13.81
0.3	7	0.3	0	6.88	1	21.3	0.01	9.26	1	22.38	0.06	8.51	1	22.04
0.3	7	0.7	0	4.19	0.8	16.44	0.01	6.06	0.8	17.29	0.02	6.65	0.8	17.56
0.7	3	0.3	0.03	10.28	0.6	12.52	0.18	1.38	0.33	4.68	n.a.	n.a.	n.a.	n.a.
0.7	3	0.7	0.05	5.47	0.6	9.99	0.15	7.12	0.4	8.49	n.a.	n.a.	n.a.	n.a.
0.7	5	0.3	0.12	4.55	0.4	7.05	0.29	4.26	-0.33	-1.52	0.3	-1.49	-0.5	-6.54
0.7	5	0.7	0.05	4.74	0.6	9.45	0.18	5.99	0.4	7.83	0.25	6.8	0	3.68
0.7	7	0.3	0.14	0.14	0.75	9.09	-	-	-	-	0.39	2.59	0	1.35
0.7	7	0.7	0.09	6.01	0.4	7.93	0.22	3.31	0.2	4.12	0.28	2.64	0	1.37

Table 5.8: Routing case - random map

Instance			$\alpha = 1$				$\alpha = 2$				$\alpha = 3$			
p	$\overline{D}$	m_d	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)
0.3	3	0.3	0.05	8.13	0.4	8.13	0.09	9.32	0.6	9.32	n.a.	n.a.	n.a.	n.a.
0.3	3	0.7	0.02	6.77	0.4	6.77	0.05	7.66	0.4	7.66	n.a.	n.a.	n.a.	n.a.
0.3	5	0.3	0.03	14.41	0.6	14.41	0.05	15.72	0.6	15.72	0.1	15.99	0	15.99
0.3	5	0.7	0.01	10.51	0.6	10.51	0.02	13.03	0.4	13.03	0.04	13.54	0.4	13.54
0.3	7	0.3	0.02	16.45	0.8	16.45	0.04	18.53	0.8	18.53	0.1	18.46	1.2	18.46
0.3	7	0.7	0	12.61	0.6	12.61	0.02	15.29	0.8	15.29	0.03	16.86	1	16.86
0.7	3	0.3	0.08	10.29	0.4	10.29	0.2	7.52	0	7.52	n.a.	n.a.	n.a.	n.a.
0.7	3	0.7	0.08	9.51	0.4	9.51	0.15	8.29	0.6	8.29	n.a.	n.a.	n.a.	n.a.
0.7	5	0.3	0.11	5.33	0	5.33	0.26	-5.63	-1	-5.63	0.32	-18.31	-2.5	-18.31
0.7	5	0.7	0.08	5.75	0.4	5.75	0.14	6.14	0	6.14	0.22	2.67	0.2	2.67
0.7	7	0.3	0.14	20.26	0	20.26	-				0.2	-0.3	-2	-0.3
0.7	7	0.7	0.09	10.17	0.2	10.17	0.16	7.84	0.2	7.84	0.2	7.64	-0.2	7.64

Table 5.9: Fleet size case - cluster map

Instance			$\alpha = 1$				$\alpha = 2$				$\alpha = 3$			
p	$\overline{D}$	m_d	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)	gap (%)	Δ-Rout (%)	Δ-Fleet	Δ-Tot (%)
0.3	3	0.3	0.02	8.79	0.4	8.79	0.1	10.15	0.6	10.15	n.a.	n.a.	n.a.	n.a.
0.3	3	0.7	0.03	6.21	0.2	6.21	0.07	7.9	0.4	7.9	n.a.	n.a.	n.a.	n.a.
0.3	5	0.3	0.01	5.97	0.2	5.97	0.05	9.48	1	9.48	0.2	9.36	0.6	9.36
0.3	5	0.7	0	4.25	0.2	4.25	0.03	6.95	0.4	6.95	0.08	7.13	0	7.13
0.3	7	0.3	0.01	7.25	0.6	7.25	0.03	9.37	0.8	9.37	0.15	9.65	0.8	9.65
0.3	7	0.7	0	4.91	0.4	4.91	0.02	6.32	0.6	6.32	0.06	7.29	0.4	7.29
0.7	3	0.3	0.13	8.56	0	8.56	0.3	9.11	0	9.11	n.a.	n.a.	n.a.	n.a.
0.7	3	0.7	0.15	5.81	0.2	5.81	0.26	7.38	-0.2	7.38	n.a.	n.a.	n.a.	n.a.
0.7	5	0.3	0.27	2.75	-0.75	2.75	0.35	7.52	0	7.52	0.36	3.07	-1	3.07
0.7	5	0.7	0.12	4.55	0	4.55	0.28	6.65	0	6.65	0.35	5.78	-0.2	5.78
0.7	7	0.3	0.22	9.61	-0.25	9.61	0.36	-4	-1	-4	0.42	11.5	-0.5	11.5
0.7	7	0.7	0.18	5.83	0.2	5.83	0.31	3.11	0	3.11	0.36	5.36	-0.4	5.36

Table 5.10: Fleet size case - random map

Chapter 6

Territory Design

6.1 Introduction

Transport companies often face the tactical problem of designing a strategy to effect regular deliveries in a service area over a certain horizon. In order to make solving the daily problem easier, and to increase the drivers' familiarity with particular areas, companies wish to pre-assign customers, or sectors of the service area, to drivers. This way, every day, a driver visits the same customers and drives within the same area. This is desirable to enhance both quality of service and efficiency (Smilowitz et al. [2013]). Increasing the consistency in service is becoming more and more important to transportation providers. In fact, due to a strong and increasing competition, they are forced to shift their attention to customers' satisfaction. On the other hand, as noted in Wong and Beasley [1984], having the same driver visiting the same customers regularly can significantly increase efficiency. Drivers become more familiar with customers, learn shortcuts, traffic and road conditions. This can reduce service and travel time, as well as administrative and operational costs.

One common approach (Wong [2008]; Ríos-Mercado and Fernández [2009]; Jarrah and Bard [2012]) is to split the service area into several sectors (or clusters of customers) called *territories* (or *districts*), to be pre-assigned to drivers. This implicitly achieves consistency and decreases the complexity of the routing problem in future daily operations (Wong and Beasley [1984]). We refer to this type of approach as Territory Based Routing (TBR).

One drawback of TBR approaches is the loss in operational and routing flexibility. The routing plans obtained every day are indeed sub-optimal. Moreover, an important factor often overlooked in the literature, the routes might be unbalanced in terms of workload, which might not be desirable for a company. An analysis on the trade-off between flexibility and workforce management is presented in Smilowitz et al. [2013].

The process of dividing the service area into sectors is referred to as *districting*,

and the class of problems studying the design of efficient territories is called Districting Problems (DPs) or Territory Design Problems (TDPs). These types of problem have already been considered in the literature (Ríos-Mercado and Fernández [2009]; Jarrah and Bard [2012]; Zhong et al. [2007]). However, districting is not the only approach considered to improve service consistency. Some authors have proposed to use an approach based on "master plans". In this approach, a standard VRP is solved over all known customers. The route for a given day is then given by simply skipping those customers that do not require service. See, for example, Groër et al. [2009]; Sungur et al. [2010].

Even though there is not a unified approach, the standard methodology for TDPs does not consider the routing component of the problem. The most common methods are variation of the location-allocation algorithm (Kalcsics [2015]). This works by first determining the territories centres (seed customers or simply geographical points) and then assigning the basic units (geographical areas or customers) to the centres. The focus is on how to assign the customers in such a way that some planning criteria, such as contiguity and compactness of the territories, are met, and the territories are balanced with respect to some measures. However, a few recent works (Schneider et al. [2014]; Wong and Beasley [1984]; Sungur et al. [2010]; Smilowitz et al. [2013]) have proposed to integrate the routing component of the problem in the design of the territories. We refer to this type of approach as *routing-based*.

However, only in Smilowitz et al. [2013] is a complete integration of the districting and routing components proposed. In Schneider et al. [2014]; Wong and Beasley [1984], the routing is used in two-stage algorithms as the base for an improved assignment phase. Moreover, to the best of our knowledge, only in Schneider et al. [2014], and partially in Sungur et al. [2010], are complex routing constraints, such as time windows, considered. In particular, we highlight Schneider et al. [2014], where the authors propose a numerical analysis of the impact that time windows have on the design of territories for routing operations. As highlighted in Schneider et al. [2014], when considering time windows, or other complex operational constraints, the routing flexibility becomes paramount in the design of territories. This is because these constraints strongly affect the routing decisions and should not be ignored in the territory design phase. One other key factor is the balance of the territories with respect to a user defined measure (e.g., routes' duration). In the vast majority of papers dedicated to TDP, as we previously mentioned, the routing decisions are ignored. Therefore, one can only estimate such measures. In many cases though (e.g., routes' duration), reliable estimates are very hard to obtain. Routing-based approaches allow for more freedom in the definition of such measures and make it possible to compute the measure exactly. The literature on routing problems with balanced routes is not

very developed. To the best of our knowledge there is no routing-base method proposed in the literature concerning TDPs that considers balanced routes. Moreover, no previous work has analysed the effect of balance requirements in territory design.

The problem addressed in this chapter is motivated by a real-world application from a grocery delivering company operating in Australia. We present a general approach which we extensively analyse. The goal is not only to solve the problem at hand, but also to investigate how much different factors affect the design of effective territories. Our analysis is similar in spirit to the one presented in Schneider et al. [2014]. Some of the questions we address overlap with the goals in their work. However, the methodology adopted is substantially different.

Our goal is to investigate the following matters:

1. The impact of time windows and balance requirements on the design of effective territories.

2. The ability of a routing-based approach to balance the daily routes throughout the whole horizon.

3. The impact of other factors, such as demand variation, on the effectiveness of a TBR approach.

To address these questions we develop a heuristic based on the Adaptive Large Neighbourhood Search (ALNS) scheme (Ropke and Pisinger [2006]). Our method is simple and easily extendable to richer routing problems. The proposed method can also be used when territories are already available. This could be seen as the second phase of a TDP.

The contributions of this chapter are several. First, we propose a simple but effective approach that can be used to solve TDPs. Notably, this integrates routing and districting decisions in the same model, allowing for a simultaneous optimisation. To the best of our knowledge no other method in the literature combines these two aspects for territory design problems. Secondly, we provide real-world based data that we use to compare with the previous method used by our industrial partner and test our algorithm. Finally, we provide an analysis that attempts to quantify what factors are important when solving a TDP. In particular, we analyse how time windows and balance requirements affect the territories effectiveness. Furthermore, as noted in Schneider et al. [2014], it is also interesting to see how TBR approach performs on a real-world problem which is not embedded in a Euclidean plane. To the best of our knowledge, no previous work has proposed a routing-based approach able to directly consider operational constraints and balance requirements.

This chapter is organised as follows: first, in Section 6.2, we describe the real-world application that motivated this study. In Section 6.3, we provide a general formulation of the problem. Then, in Section 6.4, we review the relevant literature and illustrate the advantages of having a routing-based formulation. The solution method is presented in Section 6.5. We present an extensive numerical analysis in Section 6.6. Conclusions are drawn in Section 6.7.

6.2 Motivation and Previous Work

In this section we present the real-world application that motivated this work. Our industrial partner is a grocery delivering company operating in Australia. Each distribution centre (DC or depot), every day, has to satisfy the demand of a set of customers. Not all customers require service every day. Since customers are associated with only one DC, an instance of the problem is defined by a DC. Each customer can only be visited during a specific time window, which is the same every day. Waiting at the customer location is allowed. The depot has a time window, defining the earliest start and latest return time allowed for the routes. Each day, a fleet of vehicles has to satisfy the demands of the customers. The vehicles cannot visit the depot to replenish. It is desirable to have, every day, routes of a similar duration, to reach some fairness amongst drivers. We have at our disposal a week's worth of data for several distribution centres. The goal is to design a cost effective routing plan, based on the available data, which also increase service consistency and balance between daily routes.

In a previous collaboration, the problem was solved by using an industrial solver (Kilby and Verden [2011b]) in a master-route fashion. The idea was to design the master routes by solving a routing problem including all customers at once. This implicitly defines a routing plan for the whole horizon and assure drivers are permanently assigned to customers. However, balance requirements were not considered at that time. We describe this approach in more details in Section 6.6.4. One goal that motivated the work presented in this chapter is to study the benefits of adopting a more flexible routing approach, since the master routes approach favours consistency over routing efficiency. A second goal is to include balance requirements in the approach. In this particular case, the measure to be balanced is the routes' duration. In other routing applications, different measures might be considered. We elaborate on this in Section 6.3.1.

6.3 Problem Formulation

We now present our formulation of the problem, upon which we build a general approach to TDPs. Moreover, we look at the advantages of including the routing decisions in the model.

Let us start with the formal description of how we model the problem and what we define as a solution. This fits our application and also serves as the framework we use in the computational analysis presented in Section 6.6. In the following we assume a fixed homogeneous fleet. This assumption is not true for all DCs in the application considered. However, in order to focus our attention to the districting component of the problem and to facilitate future use of the provided data, we decided to consider only this case.

We consider a time horizon made of D days (or, more generally, periods) of operations. We have a set of customers $C = \{1, \dots, N\}$, a single depot, denoted by 0, and a set F of homogeneous vehicles of capacity Q. We have available two matrices defining the distances and travel times between any pair of locations. We do not assume symmetry in the matrices (in fact they are not in the problem considered for the experiments). Each customer may place some requests during the horizon. We denote by $C_d \subseteq C$ the set of customers which require service on day d. Therefore, the total number of requests over the horizon equals $\sum_d |C_d|$. For each customer $i \in C$, we denote by $[e_i, l_i]$ its time window and by s_i the service time needed to perform necessary operations at the customer's location. The time windows of the depot is denoted by $[e_0, l_0]$.

Loosely speaking, the goal is to partition the set C in $|F|$ sets (or territories), denoted by $\{T_v\}_{v \in F}$, so that a vehicle/driver can be assigned to a unique territory[1]. Ideally, on each day d, a vehicle would only visit customers in its territory. However, this might be impossible (due to capacity constraints for example). Hence, we take a soft approach and try to increase consistency by penalising the "violations" of the partition as we describe later in the section. Moreover, each day, we want to balance the routes with respect to a given measure. For now, we leave the definition of a measure as general as possible. Later in the section we elaborate on the actual measures we consider.

Let us introduce some notation. We denote by R_d the set of all possible routes for day d and define $R := \bigcup_d R_d$. We assume $|F|$ copies of the empty route are included in R_d with the convention that, if a vehicle is assigned an empty route then such vehicle is idle on day d. The measure to be balanced, denoted by μ, is a function

[1]Note that we use the set V also to index and represent the territories. This is because the number of vehicles/drivers and territories is assumed to be the same.

$\mu : R \to [0, \infty)$. Given a route $r \in R$, we denote by $d(r)$ the day associated with the route and by $\mu(r)$ its measure.

A solution is composed by:

S.1 a set of routes $S_d \subseteq R_d$, one route for each vehicle each day (i.e., a routing plan), satisfying capacity and time window constraints;

S.2 a map ϕ, from $\bigcup_d S_d$ (the routing plan) to F, so that, for each day, the correspondence from S_d to F is one-to-one.

We first want to point out that we do not impose that all requests have to be satisfied. Therefore, a request may be unrouted, i.e., not visited by any route. In this case, a penalty is added to the objective. Let us now focus on the second part of our definition. A solution implicitly defines the territories, as we now explain. Assume we have a solution. We can label and identify territories with indexes $v \in F$. Using the same notation of the route map, we define, for each $i \in C$,

$$\phi(i) = \arg\max_{v \in F} \left| \{ r \in \bigcup_d S_d : \phi(r) = v \} \right|$$

where $\phi(i)$ represents the territory to which a customer $i \in C$ is assigned. In other words, i is assigned to the territory which is associated with the highest number of routes that visit i. From now on, we will write $\phi(i)$ to refer to the territory associated to i by the solution under consideration[2]. Summarising, a solution implicitly defines the territories, and, each day, all $|F|$ routes are assigned to different territories. Consequently, each territory has $|D|$ routes mapped into it, one for each day. When a driver (or vehicle) is assigned to a territory, it is also assigned to all routes associated with that territory.

In order to formally describe the costs, we define the unbalance $\Delta_d(\mu)$ of a solution s for a day d, with respect to the measure μ, as

$$\Delta_d(\mu) = \frac{\max_{r \in S_d} \mu(r) - \min_{r \in S_d} \mu(r)}{\min_{r \in S_d} \mu(r)}.$$

Given a tolerance $\rho \geq 0$, we further define

$$\Delta_d^\rho(\mu) = \begin{cases} \Delta_d(\mu) & \text{if } \Delta_d(\mu) > \rho \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

Therefore, we accept an unbalance in the routes as long as it is less than ρ. The

[2]The solution will be clear in the context.

cost of a solution s is denoted by $f(s)$ and obtained as a weighted sum of the following components:

1. Distance Cost: the total distance travelled throughout the horizon;
2. Territory Cost: the number of violations of the partition. A violation is defined as a route r visiting a customer i such that $\phi(i) \neq \phi(r)$;
3. Balance Cost: the sum over the days of the quantities $\Delta_d^\rho(\mu)$;
4. Outsourcing Cost: the number of unrouted requests.

We assume the Distance Cost to always have unitary weight and we denote the weights of the Territory, Balance and Outsourcing Cost, respectively, by $\lambda_{\text{terr}}, \lambda_{\text{bal}}, \lambda_{\text{out}}$. These are assumed to non-negative real numbers. If we want to make the weights explicit, we write the cost of a solution s as $f^{\lambda_{\text{terr}},\lambda_{\text{bal}},\lambda_{\text{out}}}(s)$. The goal is to find a solution that minimises the cost.

A strategy that has proved effective is to introduce a flex zone, i.e., to exclude some customers from the partition and leave them accessible to all vehicles with no penalty (Zhong et al. [2007]). Since our goal is to study the effect of several factors on a TBR approach, to not lose generality we chose to not allow such freedom. In fact, in the real-world problem we presented in Section 6.2 this is not allowed. However, it is not hard to integrate this idea in our method. A simple heuristic would be to select the customers with the highest number of violations as the ones to be placed in the flex zone. One other idea would be to set a tolerance parameter also for the Territory Cost.

6.3.1 Balance Measures

Our definition of the balance measure is route-based. It is not the territories that have a measure, but, instead, the daily routes assigned to the territories. This is consistent with our choice of basing the territory design on the routing solution. In the computational experiments we consider two different measures: route duration and route value. We refer to the balance cost as Duration Cost and Value Cost. We still write Balance Cost if we do not need to specify the particular measure we are using. The Duration Cost penalises the difference in the duration of a route, i.e., the total time from the moment a vehicle leaves the depot to the moment the vehicle returns to it. We stress that the balance is measured on a day to day basis, not cumulatively throughout the horizon. This type of balance is often important in real applications as a company might want to have routes that are fair to the drivers. To define the Value Cost we need additional data. Assume that, associated with a

customer i and each day d, there is a value $v_{id} \geq 0$. The value of a route is defined as the sum of the values of the customers that are visited by the route. The Value Cost penalises the difference in the values of daily routes. The values v_{id} can represent real monetary values, for example if we are dealing with routing of technicians and each driver/technician is paid based on the amount of work done. However, values can serve as a proxy for other quantities, for example, a simple count of customers, or the area assigned to the customer in a Voronoi diagram.

6.4 Related Work

Our work spans across different areas of the literature. The daily routing problem we examine belongs to the class of VRPs with Time Windows (VRPTW). The literature on solution methods of routing problems is vast and its review requires a dedicated survey paper. We refer to Toth and Vigo [2014] and references therein.

Instead, we focus on past work in the area concerning TDPs. The applications of such problems vary in nature, from political and school districting (Bacao et al. [2005]; Ferland and Guénette [1990]) to routing oriented applications such as sales and service territory design (Zoltners and Sinha [2005]). A general review on TDPs can be found in Kalcsics [2015]. We focus our review on routing oriented applications.

There is no standard shared approach to TDPs. Some papers make use of continuous approximation models. The underlying idea is that, by fixing the shape of the districts, one can estimate the length of a (near) optimal route. In Newell and Daganzo [1986]; Daganzo [1984], the authors propose to use a ring radial network. A disk model is proposed in Ouyang and Daganzo [2006]. In Jarrah and Bard [2012], the authors apply a similar idea to a pick-up and delivery real-world problem. They propose to divide the service area in districts contained in rectangular regions whose dimensions have to be determined. In Hall et al. [1994], an integrated method is developed. This combines the continuous model proposed in Daganzo [1984] with a discrete model, that makes use of a generalised assignment problem and a travelling salesman problem. The paper showed that the estimate for a route's length presented in Daganzo [1984] is quite accurate.

An approach combining a disk model with the concept of Voronoi tessellation is proposed in Ouyang [2007]. The authors transform the service area by means of a conformal map so that they can use a disk model. They use a discretised dynamical system to find a non overlapping covering of the mapped area using smaller disks. These are mapped back in the original area to obtain the centres of the territories. Eventually, the original service area is divided in rectangular territories by means of

a Voronoi tessellation. The approach proposed is mathematically sound and the tests show good results.

A vast part of the literature focuses on algorithmic methods to solve discrete models. A common approach (Kalcsics [2015]) is to formulate a TDP as an assignment problem. Typically, a p-centre, or p-median, measure is used to control the dispersion of the territories and some constraints are added in order to have balanced territories with respect to other measures of interest. An example studying a real-world application can be found in Ríos-Mercado and Fernández [2009]. Here the authors consider three different measures to be balanced and require the territories to be contiguous. The model presented is a p-centre problem with multiple capacity constraints and balance constraints. The problem is solved with a GRASP-based heuristic. Other examples include Salazar-Aguilar et al. [2011]; Benzarti et al. [2013].

The authors in Haugland et al. [2007] study the problem of designing districts for a stochastic VRP. They model the problem as a two-stage stochastic program with recourse and propose a multi-start and a Tabu Search (TS) heuristic to create the districts. Instead of considering routing decisions they propose to estimate the routing cost of a district with a simpler formula. They test their methods on modified standard VRP benchmarks.

In Zhong [2003] a two-stage method is presented. The first, strategical, stage aims at grouping customers in small "cells". Some of these cells are grouped, to define what they name core areas, and assigned to specific drivers. The core areas will always be visited by the same driver. The unassigned core areas form a *flex zone*, i.e., an area that can be visited by any driver. In the second phase the routing plan is designed using the defined core areas. They show how allowing a flex zone significantly increases the flexibility in the routing phase. The results show how this approach can balance the trade-off between consistency and flexibility. Moreover, in the routing phase, the insertion cost is modified to account for a driver learning model. This aims at modelling the increased familiarity of a driver with a certain area resulting from several consecutive visits.

One of the first papers using the idea of constructing territories based on the routing solutions of sample days is Wong and Beasley [1984]. Here the authors solve a number of VRPs independently. The solutions are used to compute a matrix defining the cost of having two customers in the same area. This matrix is then used to solve an allocation problem. Even though the consistency in service is strongly enhanced, the inflexibility of their territories leads to infeasibility when the demands presents significant variation. This method can be seen as a predecessor of the method proposed in Schneider et al. [2014].

In Schneider et al. [2014] the authors propose a routing-based methods to ap-

proach TDPs. They also present an extensive computational analysis aimed at evaluating the effect of time windows on TBR approaches. They first solve a set of sample days using a TS algorithm. The solutions of these problems are then used in the territory design phase. This is a two-stage method. In the first stage some seed customers are selected and then the remaining customers are assigned to the seeds to form the territories. They use the flex zone concept presented in Zhong [2003]. The assignment is not solely based on the routing history but also includes geographical or time-windows related information. By comparing these two methods they conclude that geographical information is more important than time-related information when designing territories. Their analysis is based on modified VRP benchmarks. Their method, as well as the one proposed in Wong and Beasley [1984], differs from ours in the fact that routing problems are solved without regard to territory constraints, and then the solutions combined to form territories. There is therefore, still a degree of separation between routing and districting. Moreover, in Schneider et al. [2014], only a fixed percentage (roughly 60%) of the customers is assigned to a territory, while the remaining ones are left unassigned.

In Smilowitz et al. [2013], an analysis of the trade-off between consistency and routing for the Periodic VRP (PVRP) is presented. The authors three different approaches to solve a variation of PVRP that include consistency measures. The first is a two-stage approach where a standard PVRP is solved and then the obtained routes are assigned to drivers in order to increase consistency. The other two approaches include the workforce management in the objective. A TS heuristic is used to solve the problems. Numerical tests show that, by simply including consistency measures in the objective, their model is able to produce solutions that can balance routing efficiency and service consistency. Conversely, optimising over the distance and then later assigning the routes does not always produce good results. The methodology presented in Smilowitz et al. [2013] is similar to ours in spirit. However, they do not introduce the concept of territories and the problem analysed is of a different nature. Although the cited papers also study routing in the presence of territory constraints, the approach in the present paper differs markedly, integrating routing and territory design in the heart of the algorithm.

In Haughton [2008], the author presents a comparison between an exclusive territory assignment approach and a daily re-optimisation of the routing plan. An intermediate option is considered, consisting in assigning teams of drivers to territories. This aims at increasing the routing flexibility without decreasing the consistency of service. Numerical experiments are carried out to analyse the effect of the demand variation, the fleet size and the vehicles' capacity on the efficacy of the territory approach.

In Groër et al. [2009], the authors introduce the Consistent VRP (ConVRP). This is a multi-period routing problem where the objective is to have a customer visited by the same driver at approximately the same time every day the customer requires service. The authors develop a two-stage approach based on the record-to-record algorithm. Their method is based on a precedence principle, i.e. it builds a set of master routes that are used as templates for all day routes. A modified capacity and time limit for the master routes have to be set in order to not build routes that are infeasible when applied to the specific days or with a large slack in their capacity or time use. These two parameters turn out to be key for the quality of the routing plan produced. Therefore, occasionally, the daily routes are built to evaluate the slack or violation of the true time and load limits. Based on these checks, the parameters in the master problems are updated. The rationale behind their choice is the hope that, by having a fixed schedule, the customer will be visited at roughly the same time every day. The method achieves a good level of consistency with an acceptable increment in the total time travelled. However, the number of vehicles is considerably increased when compared to a daily optimisation.

A master-plan approach is also used in Sungur et al. [2010]. The problem studied therein is a stochastic multi-period VRP with soft time windows. The stochasticity comes from having uncertain service time and probabilistic customers. The master-plan idea is modified to fit their problem by using robust optimisation to deal with the uncertainty in the service time and by dropping customers with low frequency of occurrence. The master-plan is then used as a starting point to re-optimise every day's routes once the uncertainty is revealed. Their algorithm improved the results of Groër et al. [2009].

Districting problems are also studied in the arc routing literature. Typical applications are the design of districts for waste collection (Kim et al. [2006]) or winter gritting operations (Perrier et al. [2007]). In Mourão et al. [2009], the Sectoring Arc Routing Problem is defined. This is a general abstraction of the typical territory problem faced in arc routing. The seasonality in these problems is much stronger than in the problem we are considering. In fact, such problems are usually solved for a single day of operations. We do not focus on arc routing applications and refer the interested reader to Mourão et al. [2009].

Lastly, we briefly focus on the literature concerning balance in routing applications without regard to territories. The measures to be balance vary from application to application. In Jozefowiez et al. [2007], the authors consider the standard VRP including a simple balance measure in the objective function. The balance of routes is measured simply by taking the difference between the longest and the shortest route. Similarly, in Mandal et al. [2015], the authors study the mixed capacitated general

routing problem including the same balancing measure. In Lee and Ueng [1999], the measure balanced is the working time of the drivers. One other example is the workload of the vehicles, intended as route total demand, (Ribeiro and Ramalhinho Dias Lourenço [2001]). A similar measure is considered in a multi-period problem in Mourgaya and Vanderbeck [2007]. Other references can be found in Jozefowiez et al. [2008]; Matl et al. [2017]. We also highlight the paper by Drexl (Drexl [2012a]), where the author compares the state of the art of commercial routing solvers and of the literature on rich routing problems. The author points out how balance requirements are important in practice, but quite overlooked in the literature.

6.4.1 Routing- vs Cluster-based Formulations

In this section, we want to compare our formulation of the problem with the typical formulation used to solve TDPs in the literature.

The first difference is the fact we consider a time dimension, in the form of a number of scenarios (days of operations in our case). The same choice was adopted in Schneider et al. [2014]; Sungur et al. [2010]. It is quite standard to use historical data, or scenarios, also when trying to improve service consistency (Groër et al. [2009]; Smilowitz et al. [2013]) without using the concept of territories.

Conversely, the typical approach does not include a time dimension. This makes sense, as there is no routing decision included in the models or solution methods. The most common method (e.g. Kalcsics [2015]) is to determine some seed customers and to assign the remaining customers to the seeds by means of an allocation method. We refer to these approaches as *cluster-based*, opposed to the *routing-based* approach we have taken.

The advantages of routing-based approaches are several. First, we can include operational constraints in the underlying routing problem. The constraints are explicitly considered through the routes. From a balance point of view, the measures can be computed explicitly using the routes. In a cluster-based approach, we do not have routes available. Therefore, operational constraints cannot be taken into account and it is not possible to compute routes' measures. A standard alternative is to assign a measure value to each customer, define a territory's measure as the sum of the measures of customers in the territory and balance the measures of the territories in the allocation problem. This approach works well for problems such as political districting (Kalcsics [2015]), where there is no concept of time and the measures are geometric in nature (e.g., density of population). This has the advantage of being computationally easy, but has several drawbacks when applied to routing applications. For example, it is not trivial to estimate the duration of a route, especially if

time windows are considered. Even in the case of route value, the demand variation could make the estimate far from reality, resulting in very unbalanced territories. Consider, for example, a situation where some customers order very rarely throughout the horizon. By assigning all of these customers to one territory, the measure estimated in a cluster-based approach might be very high. However, the real measure of a daily route is not reflected by the territory measure, as these customers might order on different days. One advantage of including routing decisions in the model, is that one can precisely compute the routes measure, and, if the sample days are representative enough of the horizon, reach a better balance.

6.5 Solution Method

In defining a solution method, our goal was to design a relatively simple algorithm that would be easy to extend to other operational constraints but nonetheless high-performing. We refer to our algorithm as Routing Based Territory Approach (RBTA).

The method we propose is based on the Adaptive Large Neighbourhood Search (ALNS) algorithm (Ropke and Pisinger [2006])[3] . ALNS has been proved to be a very efficient and flexible technique when applied to routing problems (Kilby and Verden [2011b]). Our method differs from the ALNS of Ropke and Pisinger [2006] in several aspects. We now focus on the various components of our method. Later in the section we will provide the pseudo-code of our algorithm.

Initialisation. We need to create an initial solution to start ALNS. We use several methods to build a solution and we take the best one produced. We use a variation[4] of the I1 insertion heuristic (Solomon [1987]) described in 3.3.1. We created two versions of the I1 heuristics, different only in term of seed selection: the first selects the furthest customer, in terms of travel time, from the depot. The second selects the customer with lowest l_i. Moreover, we create other candidate initial solutions by using some of the repair methods described later in this section.

Destroy Heuristics. We consider the *random*, *worst* and *Shaw* heuristic for removing customers as described in Ropke and Pisinger [2006]. For each of these we create a day-by-day and a horizon version. The day-by-day version selects one day at the time and applies the original heuristic. The horizon version considers the whole horizon. Therefore, once a customer i is selected, it will be removed from its current route on all days. This gives more opportunities for the insertion heuristic to decrease

[3]For a presentation of ALNS see Section 3.4.1.
[4]For this method only the Distance Cost is considered.

the Territory Cost. In particular, for the *worst* heuristic, the score associated with a customer is a simple average of the scores of all days (see Ropke and Pisinger [2006] for more details). We also introduce one new destroy operator, that we name *route-removal*. This selects and removes a random route, where the probability of a route to be chosen is inversely proportional to its length. Moreover, it removes other random customers, in order to create space for the removed customers in the other routes. Both a day-by-day and a horizon version are implemented.

Repair Heuristics. The same idea is applied to the insertion heuristics. We consider the greedy and k-regret heuristic with $k = 2, 3$ (see Ropke and Pisinger [2006]). Just as we did for the destroy heuristics, we define a day-by-day and a horizon version. The day-by-day applies the original method to each day in a random order. The horizon version considers an insertion of a customer i in a route r over the set of days where customer i requires service and is not already in another route. The score of an insertion is the sum of the day scores. The horizon version is slower that the day-by-day version, therefore we decided to use the horizon version only for the greedy and 2-regret heuristics. However, experiments show this does not result in a loss of efficiency.

In the day-by-day version, days are considered in a random order to give the possibility of changing the route of a customer. Consider a customer that has a "natural" route on day one, but on other days may be closer to other routes. If day 1 was always considered first, the customer would tend to end up on the route it was close to on day one. By considering days in random order, the customer may already be assigned to a different route on a number of days by the time day one is considered. The territory penalty would then encourage it to be assigned to that new route even on day one.

Route Mapping Operator. In order to have a faster algorithm we decided to fix the map ϕ (part S.2) of the current solution throughout the algorithm. We number the routes progressively from 1 to $|F|$, therefore assuming that route r is assigned to territory r. However, this creates some potential limitations. Consider the following scenario: by swapping all visits of two routes r_1 and r_2 on the first day of the current solution s_{curr} we obtain a different solution s' which is optimal for the problem. In fact, even if the routing plan of s' has not changed, swapping two routes can decrease the Territory Cost since the map ϕ is fixed. Unfortunately, it might be hard to reach s' from s_{curr} with the described destroy and repair heuristics. This effect is more pronounced in the initial stages of the algorithm.

To overcome this issue, we define a solution operator, which we call *route-mapping*.

This cannot be classified as a destroy or a repair heuristic. The idea is very simple: given the the map ϕ is fixed, we aim at swapping entire routes' allocation (just like in the above example), in order to improve the Territory Cost. The way we reorder the routes is dictated by the solution of a MIP model and it is done in a day-by-day fashion. Fix a day $d \in D$. For each route $r \in S_d$ and each territory index $v \in V$ we introduce a binary variable x_{rv} representing whether r is mapped to v. We define the w_{rv} as the number of violations we would create if we were to assign r to v. However, we count only the violations that occur on days we have already reordered. This is because we do not want to account for future days that we will reorder later. Therefore, the optimal mapping is given by the solution of the following model:

$$
\begin{aligned}
&\min \sum_{r,v} w_{rv}\, x_{rv} && \\
&\text{s.t.} \sum_{t} x_{rv} = 1 && \forall\, v \in V \\
&\quad \sum_{r} x_{rv} = 1 && \forall\, r \in S_d \\
&\quad x_{rv} \in \{0,1\} && \forall\, r \in S_d, v \in V
\end{aligned}
$$

The model is very simple. The two sets of constraints mean the map is one-to-one. The cost is the total number of violations created by the new map. Note that this is an Assignment Problem. This type of problem is known to be totally unimodular. Therefore, solving such a model is very fast and does not slow down the algorithm. If we were to solve the mapping problem over the whole horizon at once, the model would be more involved and would require more time to be solved, without improving the efficiency of the operator significantly. Therefore, we opted for a day-by-day version. Once we have a solution of the model, for all variables $x_{rv} = 1$, we move all visits of r to the route assigned to territory v. We use this operator at the end of every segment.

Cost Computations. We chose to not consider the Territory Cost as it is, but to increase λ_{terr} from zero to its real value during the algorithm. In other words we do not use the real cost $f^{\lambda_{\text{terr}},\lambda_{\text{bal}},\lambda_{\text{out}}}(\cdot)$ in the ALNS algorithm, but we use a modified weight $\widetilde{\lambda}_{\text{terr}}$ for the Balance Cost. We denote the resulting cost function with $\widetilde{f}$, i.e. $\widetilde{f}(\cdot) = f^{\widetilde{\lambda}_{\text{terr}},\lambda_{\text{bal}},\lambda_{\text{out}}}(\cdot)$. Let us denote by $it_{\max}$ the maximum number of iterations[5]. At the end of each segment, we set $\widetilde{\lambda}_{\text{terr}} = \lambda_{\text{terr}} \frac{it}{it_{\max}}$ where it is the number of iterations

[5]Since we are using SA on top of ALNS, one can compute $it_{\max}$ given the initial temperature, the cooling schedule and the maximum number of iterations imposed to ALNS.

completed so far. We found this to perform better when compared to a standard computation of the cost. However, at every iteration we have to evaluate the real cost of the newly sample solution and keep a "real best" solution in addition to the best solution used in standard implementations of ALNS.

A more subtle problem is how to manage the Balance Cost during the repair phase. Consider the duration balance constraint. Since the Balance Cost penalises the gap between routes' duration, a bad insertion early in the repair phase might be highly rewarded only because it brings the durations of the routes closer. This represents a concrete issue for the repair heuristic. In fact, the destroy heuristic often creates unbalanced solutions and we wish to avoid the repair heuristic being driven to sub-optimal insertions by the Balance Cost. We propose three possible ways of modifying the Balance Cost computation in order to remedy this issue. In the first, we simply ignore the Balance Cost during the repair phase. In the second, we multiply the Balance Cost by the fraction of customers already inserted in the solution. Finally, in the third, we only consider the Balance Cost component during the last insertions. A preliminary computational test showed that the second option is the one that performs the best. In details, when a new insertion is evaluated, if there are still m customers to be inserted, the weight of the Balance Cost in the cost computation is set to $\lambda_{\text{bal}} \frac{|C_d|-m}{|C_d|}$. We stress that this is only done within the repair heuristic. The cost of the repaired solution is computed using the original λ_{bal}.

The pseudo-code of our algorithm is given in Algorithm 6.

6.6 Computational Analysis

In this section we perform an extensive computation analysis with several goals in mind: analysing the trade-off between flexibility and consistency, comparing our approach with the previous solution method for the application described in Section 6.2, and studying and quantifying the impact of several factors on the design and effectiveness of our TBR approach.

In Section 6.6.1 we test RBTA on the standard set of benchmark instances proposed by Solomon (Solomon [1987]) in order to validate its quality as a VRPTW routing solver and its performance when a Balance Cost is considered.

Throughout all the experiments we use the ALNS parameter configuration used in Ropke and Pisinger [2006] as this has already been fine tuned and is quite hard to beat. If not otherwise stated, we set the maximum number of iterations to 50000. Moreover, we set λ_{out} to a value that dominates the other costs, to avoid having unrouted customers.

Algorithm 6 RBTA

create an initial solution s
apply the route-mapping operator to s
set $s_{\text{best}} = s$, $s_{\text{curr}} = s$, $s_{\text{realbest}} = s$
set all weights of destroy and repair heuristics to 1
set $iteration = 0$
repeat
  for M iterations **do**
    set $iteration = iteration + 1$
    sample a destroy heuristic H^- and a repair heuristic H^+ according to their weights
    set $s' = s_{\text{curr}}$
    use H^- to remove customers from s'
    use H^+ to insert the removed customers to s'
    if $f(s') < f(s_{\text{realbest}})$ **then**
      set $s_{\text{realbest}} = s'$
    end if
    if AcceptanceCondition(s', s_{curr}, $iteration$) **then**
      set $s_{\text{curr}} = s'$
      if $\widetilde{f}(s') < \widetilde{f}(s_{\text{best}})$ **then**
        set $s_{\text{best}} = s'$
      end if
    end if
  end for
  apply the route-mapping operator to s_{curr}
  set $\widetilde{\lambda}_{\text{terr}} = \lambda_{\text{terr}}\, iteration \,/\, it_{\max}$
  update weights of destroy and repair heuristic
until stopping condition is met

6.6.1 Quality Assessment

In this section, we assess the quality of RBTA. We follow the same approach taken in Schneider et al. [2014] and run the algorithm one the well-known Solomon instances (Solomon [1987]). Since RBTA was not designed to minimise the number of vehicles, we start our solution with the best number of vehicles known in the literature as done by the authors in Schneider et al. [2014]. We recall that the 56 Solomon instances are split in 6 groups. The first distinction is on the customers disposition, which can be random (R), clustered (C) or mixed (RC). Moreover, instances in groups R1, C1 and RC1 have low capacity vehicles and a short scheduling horizon $l_0 - e_0$. On the other side, groups $R2, C2$ and $RC2$ have significantly longer scheduling horizon and vehicles with higher capacity. This results in solutions for each instance in the former groups having a higher number of vehicles performing shorter routes, when compared with solutions of the latter groups. Since these instances do not have a

multi-day structure, our algorithm reduces to the ALNS classic scheme, with the addition of the route-removal operator we introduced and the different initialisation process.

To measure the quality of our results we use the cumulative number of vehicles (CNV) and cumulative travelled distances (CTD). These are widely used measures when assessing the quality of a VRPTW solver. We obtained a CNV of 405 and a CTD of 57673.04, which corresponds to a gap of 0.0% and 0.85%, when compared to the best known results as reported on the Transportation Optimization Portal of SINTEF Applied Mathematics[6].

We also want to test the quality of the solutions when balance requirements are involved. We consider all Solomon instances and set the number of vehicles to the best-known results plus one. We first run RBTA without including any Balance Cost. Then, for both the Duration and Value Cost, we run RBTA twice on each instance, with a low value for λ_{bal}, so that the Duration (or Value) Cost is comparable with the Distance Cost, with a very high value for λ_{bal}, so that the Duration (or Value) Cost dominates the Distance Cost. In both cases we set the tolerance $\rho = 0.15$. We set $\lambda_{\text{bal}} = 10^5$ in the "high" case. For the low case the value of λ_{bal} varies from group to group. We chose values in order to make the Balance Cost lower than the Distance Cost but not insignificant to the problem.

λ_{bal}	R1	C1	RC1	R2	C2	RC2	all
Distance Cost							
none	1189.57	1090.89	1363.79	903.44	825.45	1045.98	999.37
low	1223.59	1145.09	1440.17	909.36	839.91	1062.91	1028.41
high	1282.98	1263.22	1475.12	933.15	857.45	1096.88	1086.69
Route Duration Gap (%)							
none	1.97	6.67	2.67	0.39	inf	0.66	inf
low	0.32	0.24	0.21	0.12	0.15	0.17	0.21
high	0.14	0.14	0.15	0.11	0.13	0.13	0.13

Table 6.1: Results due to difference level of λ_{bal} when the Duration Cost is considered. The row labelled by "none" indicates that Duration Cost was not considered. If a route is empty, we mark the gap in the duration as "inf".

In Table 6.1, we summarise the results concerning the Duration Cost. For each group of instances, we report the Distance Cost and the average gap between the longest and shortest routes (RDG). The last column considers all instances. We can see that RBTA is very effective in reducing the duration gap without increasing the

[6]https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/

λ_{bal}	R1	C1	RC1	R2	C2	RC2	all
Distance Cost							
none	1189.57	1090.89	1363.79	903.44	825.45	1045.98	999.37
low	1230.2	1152.84	1431.05	913.66	863.93	1059.09	1039.89
high	1263.2	1236.43	1498.01	915.69	871.27	1068.84	1074.83
Route Value Gap (%)							
none	4.35	13.84	8.03	0.76	inf	0.66	inf
low	0.34	0.45	0.45	0.12	0.23	0.16	0.3
high	0.14	0.14	0.15	0.11	0.13	0.13	0.13

Table 6.2: Results due to difference level of λ_{bal} when the Value Cost is considered.

travelled distance too much. Moreover, when increasing λ_{bal} from a low to high value, the duration gap is consistently brought below the tolerance without a dramatic increase in the TD. In Table 6.2, we do the same thing for the Value Cost. We report the Distance Cost and average gap between most and least valuable routes (RVG). Similar conclusions can be drawn in this case, with the only difference being that the solution when no Balance Cost is considered exhibit very high gaps.

6.6.2 Test Data and Quality Measures

We have available some real data from our industrial partner. We cannot publish the original data but, in order to allow for future comparisons, we created some instances that we made available[7]. We created a set of 26 instances with various numbers of customers that we use for most of our tests, each having a one-week horizon. We refer to this set of instances as *group-A*. In Section 6.6.12, we use an additional set of 4 instances, which is denoted as group-B. More details on these instances are given in Section 6.6.3. In some sections, we modify the instances to isolate and analyse some factors, such as demand variation or time windows. If so, we thoroughly describe what are the modifications and how are they done. Unless otherwise stated, we set the "value" of a customer v_{id} equal to the demand of a customer i on day d.

To evaluate our results, we make use of the following measures[8], used also in Smilowitz et al. [2013]; Schneider et al. [2014].

- *Customer Familiarity* (CF): for a customer i, this is the number of days d where the route $r \subseteq S_d$ visiting i satisfies $\phi(r) = \phi(i)$, i.e., the number of days when i

[7] https://fbertoli.github.io./downloads/

[8] These are not to be confused with the balance measures. They only serve the purpose of evaluating the quality of a solution under different points of view.

is visited by the assigned driver. The CF of a solution is obtained averaging the CF of all customers.

- *Driver Diversity* (DD): for a customer i, this is the total number of drivers visiting customer i throughout the horizon, i.e., the cardinality of the set $\{\phi(r) \mid r \in \bigcup_d S_d : r \text{ visits } i\}$. The DD of a solution is obtained averaging the DD of all customers.

- *Routes Duration Gap* (RDG): for a solution this is the average daily gap in routes' duration. It is computed as $\frac{1}{|D|}\sum_d \Delta_d(\mu)$, where, in this case μ represents the duration of a route.

- *Routes Value Gap* (RVG): for a solution this is the average daily gap in routes' value. It is computed just as RDG but using the value of a route as μ.

- *Travelled Distances* (TD): the average distance travelled per day.

- *Outsourced Requests* (OR): the average number of unrouted requests per day.

- *Feasible Days* (FD): the fraction of days with no unrouted requests.

As we can see, CF and DD quantify the consistency in service of a solution and the effectiveness of the RBTA. The routing quality is measured by TD and OR. The measure FD serves both objectives. Lastly, RVG and RDG evaluate the balance in the territories.

Unless otherwise stated, we set $\lambda_{\text{out}} = 10^8, \lambda_{\text{terr}} = 10^5, \lambda_{\text{bal}} = 10^3$ and $\rho = 0.1$. This way, the algorithm tries to respect the territories as much as possible, but a violation of the territories is preferred to not visiting a customer.

6.6.3 Instances Details

In Table 6.3, we report some information about the instances in group-A. Namely: the total number of customers (N), the number of territories ($|F|$), the average of customers per day (req average), the average number of days a customer require service (frequency), the average of the TW width divided by the scheduling horizon, i.e. the TW of the depot, (TWW) and the fraction of customers having a time window (TWD). Note that, even though the density of time windows is very high, the width is quite large compared to the scheduling horizon. Note also that the frequency is quite high, which is also reflected in the average number of customers on each day begin close to the total number of customers.

instance	N	$\|F\|$	req-average	frequency	TWW	TWD
instance-A-1	65	3	48.1	5.0	0.6	1.0
instance-A-2	86	4	57.1	4.0	0.7	1.0
instance-A-3	61	2	42.9	4.0	0.6	1.0
instance-A-4	34	2	19.6	4.0	0.7	1.0
instance-A-5	110	4	83.7	5.0	0.6	0.7
instance-A-6	114	5	68.1	4.0	0.6	1.0
instance-A-7	89	3	70.1	5.0	0.6	1.0
instance-A-8	66	3	48.4	5.0	0.7	1.0
instance-A-9	82	3	63.1	5.0	0.8	1.0
instance-A-10	32	2	22.7	4.0	0.6	1.0
instance-A-11	119	5	88.1	5.0	0.7	1.0
instance-A-12	55	3	39.4	5.0	0.6	1.0
instance-A-13	62	3	44.0	4.0	0.7	1.0
instance-A-14	57	3	41.9	5.0	0.7	1.0
instance-A-15	203	10	158.4	5.0	0.7	1.0
instance-A-16	43	3	26.3	4.0	0.7	1.0
instance-A-17	42	2	31.6	5.0	0.6	1.0
instance-A-18	49	2	37.7	5.0	0.6	1.0
instance-A-19	114	7	80.1	4.0	0.6	1.0
instance-A-20	85	3	61.4	5.0	0.6	1.0
instance-A-21	125	6	91.0	5.0	0.5	1.0
instance-A-22	86	4	59.6	4.0	0.7	1.0
instance-A-23	86	4	64.4	5.0	0.6	1.0
instance-A-24	64	3	46.0	5.0	0.6	1.0
instance-A-25	42	2	36.4	6.0	0.5	1.0
instance-A-26	94	4	67.3	5.0	0.6	1.0

Table 6.3: Group-A instances.

In order to give an idea of the customers' disposition we plot the map[9] of two distribution centres in Figure 6.1. It is possible to see that there are clusters of customers as well as some far isolated customers that make the balancing of territories not trivial.

6.6.4 New vs Old Approach

We now want to compare RBTA with the previous approach described in Section 6.2. We refer to it as Master Route Approach (MRA). As previously mentioned, using a master schedule has already been used in order to improve consistency (Groër et al. [2009]; Sungur et al. [2010]). Moreover, it has been extensively used in the stochastic

[9]Since we had to modify the original coordinates, the map is not precisely identical to the original instance but the variation is minimal.

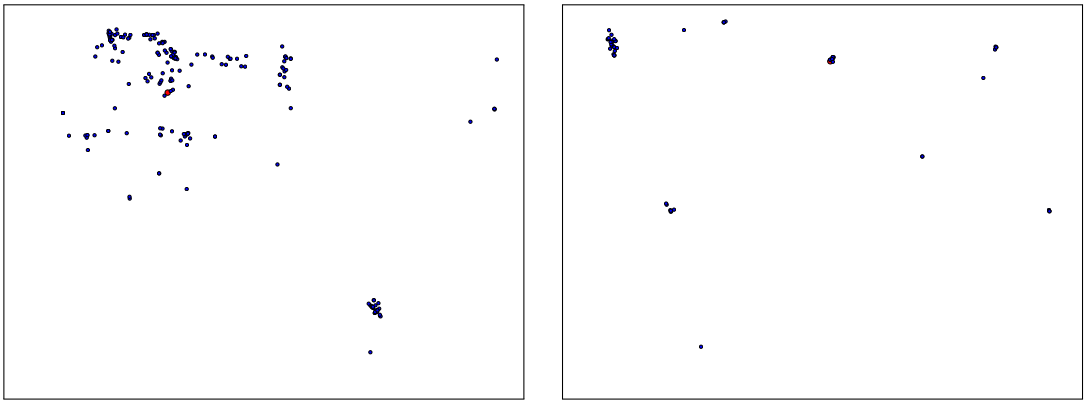


Figure 6.1: The red dot indicates the depot. Blue dots are customers location. Note that the distances and travel time between the customers are not euclidean but based on a real road network.

routing literature (Gendreau et al. [2014]). The idea is to define a single day VRPTW, called the master problem, considering all customers at once. The routes obtained by solving such problem are the master routes. From these, one creates routes for each day of the original problem by dropping customers that do not require service on the chosen day. One distinctive feature of MRA is that it considers the demands relative to each day as different commodities. Hence the master problem is a multi-commodity VRPTW with $|D|$ different types of commodities. Moreover, vehicles have $|D|$ separated compartments, one dedicated to each day, with capacity equals the original capacity of the vehicle. This eliminates the complexity of choosing and adjusting a capacity for the master problem vehicles. We have available the industrial solver that was used to implement MRA. This is an extremely flexible and efficient implementation of ALNS based on the work presented in Kilby and Verden [2011b].

We run MRA and RBTA on the instances of group-A. These are particularly suitable for a master route approach given that the time windows are wide and many customers place requests almost every day. We run both algorithms 5 times on each instance to have a fair comparison. Since MRA does not support balancing of routes we do not consider any Balance Cost in this section. We set the number of iterations of ALNS in both algorithms to 50000. In Table 6.4, for each instance, we report the average over the 5 runs of some statistics. For RBTA we report the average gap of TD when compared to MRA (Δ-TD). Hence a negative gap means RBTA improves the MRA results. It is possible to see that almost in every case the TD is improved. This is the results of the extra flexibility afforded by not enforcing a "master-route" approach. As one might expect, the improvement is more significant for bigger instances. Clearly, MRA will always produce solutions with unitary CF and DD. This

	MRA					RBTA				
instance	TD	OR	FD	CF	DD	Δ-TD	OR	FD	CF	DD
A-1	250.4	0.0	1.0	1.0	1.0	-0.15	0.0	1.0	1.0	1.0
A-2	429.3	0.0	1.0	1.0	1.0	-2.8	0.0	1.0	1.0	1.0
A-3	170.7	0.0	1.0	1.0	1.0	-0.25	0.0	1.0	1.0	1.0
A-4	134.7	0.0	1.0	1.0	1.0	-2.44	0.0	1.0	1.0	1.0
A-5	880.6	0.0	1.0	1.0	1.0	-0.4	0.0	1.0	1.0	1.0
A-6	834.4	0.0	1.0	1.0	1.0	-0.49	0.0	1.0	1.0	1.0
A-7	459.4	0.0	1.0	1.0	1.0	-7.44	0.0	1.0	1.0	1.0
A-8	722.5	0.0	1.0	1.0	1.0	-0.06	0.0	1.0	1.0	1.0
A-9	633.3	0.0	1.0	1.0	1.0	-1.04	0.0	1.0	1.0	1.0
A-10	168.5	0.0	1.0	1.0	1.0	-0.17	0.0	1.0	1.0	1.0
A-11	570.5	0.0	1.0	1.0	1.0	-0.79	0.0	1.0	1.0	1.0
A-12	1164.4	0.0	1.0	1.0	1.0	-2.44	0.0	1.0	1.0	1.0
A-13	166.3	1.0	0.0	1.0	1.0	-0.14	0.0	1.0	1.0	1.0
A-14	695.4	0.0	1.0	1.0	1.0	-0.17	0.0	1.0	1.0	1.0
A-15	938.5	0.0	1.0	1.0	1.0	0.51	0.0	1.0	1.0	1.0
A-16	834.6	0.0	1.0	1.0	1.0	-3.69	0.0	1.0	1.0	1.0
A-17	46.4	0.0	1.0	1.0	1.0	-0.37	0.0	1.0	1.0	1.0
A-18	126.0	0.0	1.0	1.0	1.0	-7.07	0.0	1.0	1.0	1.0
A-19	1659.5	0.0	1.0	1.0	1.0	-12.01	0.0	1.0	1.0	1.0
A-20	322.5	0.0	1.0	1.0	1.0	-0.81	0.0	1.0	1.0	1.0
A-21	1351.2	0.0	1.0	1.0	1.0	-17.05	0.0	1.0	1.0	1.0
A-22	520.5	0.0	1.0	1.0	1.0	-2.93	0.0	1.0	1.0	1.0
A-23	828.9	0.0	1.0	1.0	1.0	0.77	0.0	1.0	1.0	1.0
A-24	483.7	0.0	1.0	1.0	1.0	-0.2	0.0	1.0	1.0	1.0
A-25	859.4	0.0	1.0	1.0	1.0	-0.06	0.0	1.0	1.0	1.0
A-26	714.9	0.0	1.0	1.0	1.0	-0.18	0.0	1.0	1.0	1.0
average	606.3	0.04	0.96	1.0	1.0	-2.38	0.0	1.0	1.0	1.0

Table 6.4 Comparison between new and old approach.

is not necessarily true for RBTA since these constraints are “soft” in RBTA. However, RBTA produced all best solutions with unitary DD and CF, showing its ability to improve consistency. Note also that in one case, instance-A-13, MRA was not able to insert all visits in the routing plan, while RBTA always has OR equals to 0. We point out that the solver used to implement MRA is the result of a long term project that led to the development of an industrial solver which is very efficient and versatile. However, overall, RBTA proves to perform better, as the consistency is kept to the maximum possible and the routing is improved.

6.6.5 Comparison with a Two-Stage TBR Approach

As mentioned in the introduction, the closest work to ours in the literature is Schneider et al. [2014]. In this section, we compare the method presented in Schneider et al. [2014] with the one proposed in this paper. The instances used in Schneider et al. [2014] are obtained from the Gehring and Homberger benchmarks with 1000 customers. For each instance, the authors create a number (τ_1) of sample days and 50 evaluation days. The territories are created using the sample days, and their quality is evaluated using the evaluation days. The evaluation phase consists in solving each evaluation day separately, using the territories as input. That is, once the territories are fixed, we do not need to solve one big problem composed of 50 days, but we can solve 50 independent problems.

In this section, we use the same approach, i.e., we use RBTA to design territories using a set of sample days, and then we test such territories on the same set of evaluation days. Note that, in the evaluation phase, customers are already assigned to territories and therefore some shortcuts can be taken in RBTA.

In Schneider et al. [2014], the authors run experiments using $\tau_1 = 10, 50, 100$ sample days. They conclude that setting $\tau_1 = 50$ produces the best results. Therefore, we compare only with this case. However, we only use 15 sample days. The reason is that, in our method, the set of sample days is considered as a whole instance. Hence, we do not want or need many sample days. Conversely, the method presented in Schneider et al. [2014] solves each sample day separately, and therefore the number of sample days has to be quite high. Since our algorithm assumes a fixed number of territories, we first solve each of the sample days individually adding a new component to the cost function that penalises the number of vehicles used. successively, we set $|F|$ to the maximum over all sample days. Once we have determined $|F|$, we apply RBTA to the set of sample days.

In Table 6.5, we compare the results obtain by our method with the ones obtained in Schneider et al. [2014]. The table reports the average over all instance proposed in Schneider et al. [2014]. The column denoted by $|F|$ reports the average number of territories.

method	TD	$\|F\|$	OR	FD	CF	DD
Schneider et al. [2014]	10490	11.37	8.78	0.89	0.78	2.11
RBTA	12302	12.83	0.02	0.99	0.84	1.86

Table 6.5 Comparison with the TBR approach presented in Schneider et al. [2014].

The average number of territories, $|F|$, and TD are slightly higher for RBTA. This

is due to the fact that RBTA takes a number of territories as an input while the method presented in Schneider et al. [2014] minimises such number while designing the territories. However, all the other measures are improved by RBTA. Both OR and FD are close to their optimal value. Notably, the consistency measures are improved drastically. We point out that, oppositely to Schneider et al. [2014], we do not consider a flex zone.

6.6.6 Analysis Methodology

Before going into the numerical analysis, we summarise the methodology used and the naming convention to make the understanding of the following sections easier. We use the terminology No Balance case, Duration case or Value case to differentiate between different experiments that include, or not, balance requirements, and, if needed, to specify what particular balance measure μ we are using. Moreover, in order to avoid confusion, we do not report RDG for experiments including Value Cost and, vice versa, we do not report RVG for experiments including Duration Cost. However, we might report both measure for the No Balance case, so that it is possible to compare the two cases (with and without balance requirements).

In the following sections, we use the same methodology to analyse the impact of several factors. First of all, we modify the instances to isolate the factor of interest. Then we solve the instances, using RBTA, first without and then with Territory Cost. In the former case we are simply solving each day separately, an approach we refer to as daily routing re-optimisation (DRR). This approach gives us a lower bound on the routing cost that completely ignores the influence of the territory constraints. In the latter case, we are using a TBR approach. We then compare the quality measures presented in Section 6.6.2 for both approaches. For an easier comparison, for all quality measures, we report the percentage increase of TBR with respect to DRR, and the actual value[10] of TBR. This way one can easily see the losses and gains of a TBR approach with respect to DRR; that is, how the TBR is affected by the factor being considered. The percentage increases are denoted with a Δ. As an example, Δ-TD indicates the increase in TD of TBR with respect to DRR. In some cases, the percentage increase for OR and FD can be infinity. In this case we consider the average of the increases without the infinity value. However, we mark the table entry with the superscript $*$ to make clear this happened. In the following tables, each entry represents the average taken over all instances solved with the same cost configuration. Whenever we modify the instances we make clear how the average is taken.

[10]Note that for space reasons we round TD to the nearest integer.

6.6.7 Approach Validation

We now want to test the efficacy of our approach. We use an approach similar to the one proposed in Schneider et al. [2014] and used in Section 6.5 and test the obtained territories on new sample days. In more detail, given an instance, we solve the problem, as we defined it, on the available days to obtain a partition of C, denoted by T, in $|F|$ territories. We then create $\widetilde{D}$ evaluation days and test our territories on these new days to see if the territories are effective. The difference with Section 6.5 is that now we compare the DRR and TBR approaches by solving the evaluation days with both approaches, as explained in Section 6.6.6.

Next, we describe how we create new days from the available instances. Unlike Schneider et al. [2014], and like many real-world applications, we do not have explicit distributions describing the probability that a customer requires service on a given day and the demand variation. However, we use the information contained in the available instances to create new sensible days. For each customer $i \in C$, let us denote by q_{id} the demand of i on day $d \in D$. We define the probability that i requires service on a day as

$$\text{Prob}_i = \frac{|\{d \in D : q_{id} > 0\}|}{|D|}.$$

Furthermore, consider the set $\{q_{id} \,|\, d \in D : q_{id} > 0\}$ and let μ_i and σ_i be its mean and standard deviation. We assume that the demand of i is distributed according to a Gaussian random variable $\mathcal{N}(\mu_i, \sigma_i^2)$. Moreover, we assume all demand variations are independent. This is a standard assumption across the literature when using sample days (see Haugland et al. [2007]; Smilowitz et al. [2013]; Schneider et al. [2014]). Using these two distributions we can create a set of new days for each instance in group-A.

We consider all three cases, No Balance, Duration and Value. In order to apply the TBR approach, we need to solve the original instances, to obtain the territories partition T. Then we proceed to solve the new $\widetilde{D}$ days using T as input. The results are reported in Table 6.6.

It is possible to appreciate that the territory approach, TBR, increases the consistency level almost to the maximum possible, improving the CF and DD measure by more than 50 %. Moreover, the routing quality of the solution is not significantly decreased. Notably, the fraction of days with requests routed is close to 1 and the number of unrouted requests per (infeasible) day is very low. As expected, the increase in TD is higher if balance requirements are considered. Note that both RDG and RVG are kept below the accepted threshold. Therefore, the high percentage increase is misleading in this case. Overall, the proposed approach is very effect-

	No Balance		Duration		Value	
	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR
TD	4.1	570.55	12.79	727.34	10.08	703.95
OR	0.0	0.0	2.08*	0.05	0.0*	0.02
FD	0.0	1.0	-2.51	0.97	-0.57	0.98
CF	40.52	1.0	54.61	1.0	45.8	0.99
DD	-51.76	1.01	-57.03	1.0	-66.78	1.01
RDG	-	-	65.04	0.09	-	-
RVG	-	-	-	-	135.97	0.1

Table 6.6 Approach validation.

ive. The territories designed guarantee an almost optimal level of consistency and good balance over the new sample days without significantly decreasing the routing quality.

6.6.8 Number of Territories

In this section, we analyse how the number of zones affects the efficiency of TBR. In order to do so, we solve each instance of group-A with its original number of territories $|F|$, then reducing and increasing $|F|$ by one. We do this for all three cases: No Balance, Duration and Value. In Table 6.7 we report the results. The headers $-1, 0, +1$ represent whether the number of territories is reduced, original or increased.

The results show that changes in $|F|$ do not affect TD and RDG. Instead, increasing the number of territories improves the gaps under all other measures. The gaps in the feasibility measures, Δ-OR and Δ-FD, are reduced as so is the balance gap for the Value Cost. Moreover, the gain in consistency increases proportionally to $|F|$, showing that a higher number of drivers offers more opportunity for improving consistency over DRR. However, this does not mean that increasing the number of drivers is good for TBR and DRR, but only that the gaps are higher. In fact, increasing $|F|$, together with the presence of balance requirements, lead to an increased TD for both approaches. Moreover, we observe that for the original number of territories we already have a satisfactory result for the TBR approach. Indeed, OR is close to zero, all FD, CF and DD are close to 1 and both RVG and RDG are under the tolerance ρ. By looking at the actual value of all measures for TBR, one realises that increasing the number of territories over the necessary value lead only to a higher TD.

		No Balance			Duration			Value		
		-1	0	+1	-1	0	+1	-1	0	+1
TD	Δ (%)	7.3	2.8	4.1	14.9	13.8	13.1	9.6	9.4	13.9
	TBR	690.14	672.14	708.57	750.57	832.43	957.71	765.71	825.57	897.29
OR	Δ (%)	42.4	0.0	0.0	58.2*	2.8*	0.0	76.4	23.7	-10.5
	TBR	1.4	0.0	0.0	1.7	0.03	0.0	1.9	0.1	0.0
FD	Δ (%)	-19.5	0.0	0.0	-22.5	-1.8	0.0	-23.6	-2.8	1.7
	TBR	0.5	1.0	1.0	0.44	1.0	1.0	0.44	0.9	1.0
CF	Δ (%)	44.1	48.4	60.8	43.4	64.5	79.0	41.5	54.1	67.1
	TBR	1.0	1.0	1.0	0.99	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-46.5	-53.3	-57.7	-47.3	-58.4	-65.2	-48.3	-55.3	-62.6
	TBR	1.01	1.0	1.0	1.02	1.0	1.0	1.01	1.0	1.0
RDG	Δ (%)				55.9	16.05	39.8			
	TBR				0.07	0.05	0.08			
RVG	Δ (%)							210.7	175.5	34.2
	TBR							0.2	0.1	0.03

Table 6.7 Number of territories influence.

6.6.9 Balance Cost Influence

We now focus on the impact of balance requirements. In order to do so we run RBTA on all instances in group-A several times, varying the coefficient λ_{bal} for both Duration and Value Costs. The idea is to look at how the quality of the solutions change as the balance requirements becomes more and more important. We use three different cost configurations for both Duration and Value Cost:

- low: $\rho = 0.15$ and $\lambda_{\text{bal}} = 10$;
- medium (med): $\rho = 0.1$ and $\lambda_{\text{bal}} = 10^3$;
- high: $\rho = 0.05$ and $\lambda_{\text{bal}} = 10^7$;

We solve all instances with all cost configurations applying both DRR and TBR approaches. In Table 6.8, we report the results.

There are a few remarks to be made. Not surprisingly, as the importance given to the balance requirements increases, the routing quality obtained by the TBR approach decreases. This is particularly clear when looking at Δ-TD. Secondly, introducing a Balance Cost increases the gain in consistency, i.e., the consistency measure deteriorates for the DRR approach, while TBR is able to keep them at the optimal value

		No Bal.	Duration			Value		
			low	med	high	low	med	high
TD	Δ (%)	4.1	4.45	16.03	47.53	3.73	7.77	37.68
	TBR	570.57	599.1	700.9	1037.7	598.4	692.0	950.9
OR	Δ (%)	0.0	0.0*	2.08*	0.0*	0.0	18.0	30.68*
	TBR	0.0	0.02	0.02	0.11	0.01	0.05	0.14
FD	Δ (%)	0.0	-0.57	-1.37	-6.29	0.0	-2.13	1.14
	TBR	1.0	0.98	0.98	0.93	0.99	0.97	0.89
CF	Δ (%)	40.52	50.89	55.09	49.47	41.99	45.73	42.29
	TBR	1.0	1.0	1.0	0.96	1.0	1.0	0.96
DD	Δ (%)	-47.74	-52.59	-52.93	-48.91	-46.39	-50.23	-43.52
	TBR	1.0	1.0	1.0	1.16	1.0	1.0	1.17
RDG	Δ (%)	-35.7*	93.32	14.01	28.52			
	TBR	0.42*	0.19	0.05	0.05			
RVG	Δ (%)	-38.67*				62.02	145.35	133.53
	TBR	0.77*				0.29	0.11	0.05

Table 6.8 Balance cost influence.

of 1. It is interesting that, even with the strict balance requirements ($\rho = 0.05$), TBR is able to find solutions with only a few unassigned requests. Lastly, we point out that the TBR approach suffers a deterioration in consistency and a significant loss in routing quality only in the *high* case. Therefore, we can conclude that, introducing balance requirements offers more opportunities for improving consistency. However, if a very low unbalance is accepted and the weight of the Balance Cost component is very high, the TBR approach produces solutions with lower routing quality and a slightly lower consistency.

6.6.10 Time Windows Influence

We now analyse the impact of time windows. As already pointed out, a similar analysis is presented in Schneider et al. [2014]. However, the methodology adopted is different. In Schneider et al. [2014], the authors develop a constructive routing-based heuristic. They consider a set of sample days that are solved separately. In a second phase, the territories are built, using a mixture of information extrapolated from the solutions and information given by the features of the instance. They propose

several methods which exploit different types of information, such as: geographical, historical or time-window related. They draw their conclusions by comparing the results obtained with the different methods. Conversely, our model considers both the routing and districting aspects at the same time. It is therefore impossible for us to not include time windows information in the algorithm. As already explained, our methodology is to isolate the factor in the instances and then to compare all solution quality measures obtained by the DRR and TBR approaches.

We now describe how we modify the instances in group-A in order to better study the role played by time windows. We isolate two factors: time windows density (TWD): the percentage of customers having a time windows; and time windows width (TWW): the average width $l_i - e_i$. From each instance in group-A we create 13 different instances. First, we group them by TWD. The groups are:

- noTW: none of the customers has a time window;
- few: 25% of customers have a time window;
- half: 50% of customers have a time window;
- many: 75% of customers have a time window;

For each of the last 3 groups we further divide the instances in 4 groups, different in terms of TWW. These are:

- w-0: time windows have width $\overline{w} =\simeq 0.15 * (l_0 - e_0)$;
- w-1: time windows have width $\overline{w} =\simeq 0.1 * (l_0 - e_0)$;
- w-2: time windows have width $\overline{w} =\simeq 0.05 * (l_0 - e_0)$;
- w-3: time windows have width $\overline{w} =\simeq 0.01 * (l_0 - e_0)$.

Not all the widths in each group are exactly the same. There is a minor variation due to the fact that we increase the width of customers which are further away from the depot. The reason to do so is we make sure the time windows do not make a customer nearly impossible to be visited.

When increasing the time windows density, we do not assign all the time windows again, but we keep all those customers which already have a time window and select more customers to be assigned one. Therefore, each time we increase density, we only have to select 25% additional customers to which we assign a new time window. Analogously, when tightening the width, the customers having a time windows are the same. We believe this make our analysis more solid, as we are isolating the

time windows factors, while the randomness in assigning time windows is factored away.

Since we are analysing a time related constraint, in this section we only consider two cases: no Balance Cost and Duration Cost. The results are reported, respectively, in Table 6.9 and 6.10. For each value of TWW and TWD, the average is taken over all the instances associated with these fixed values; that is, one modified instance for each original instance. Note that, since varying the width does not change instances in the "none" group, there is only one element in the related columns.

The results in both tables are similar and might not be easy to read at a first look. We first focus on the routing component. The actual value of OR and FD show that, as the time windows get tighter or more dense, the instances become harder to solve. The fact that TD does not increase substantially is partly due to the fact that more and more customers cannot be inserted in any routes (i.e., OR increases). Obviously, time windows have an impact on the routing quality of the TBR. However, this is not as high as one might expect and is not proportional to the width or density. For tight or dense time windows, the percentage change of OR and FD is very small ($< 2\%$). On the other hand, for loose or few time windows, percentage changes are higher, but the actual values of OR and FD are low.

A high level of consistency is reach by TBR regardless of the time windows with the balance requirements having a slight deteriorating effect on both consistency measures CF and DD. Moreover, we observe that introducing time windows in the problem offers more opportunities to improve consistency. On the other hand, making the time windows harder to achieve has opposite effects for the No Balance and the Duration Case. In the former, the gain in consistency increases, in the latter it decreases. Finally, even though, not surprisingly, time windows have a clear impact on the value of RDG, this is not worse nor better for TBR. In other words, time windows do not affect the ability of TBR to balance the territories.

In conclusion, the impact of time windows on TBR can be summarised in the following points:

- the introduction of time windows in the problems offers more opportunities to improve consistency;
- introducing time windows and increasing their complexity affects the routing quality of the solution, though the impact is dramatic only in extreme cases;
- time windows do not affect the ability of TBR to balance the territories and reach a good consistency level.

		Δ (%)				TBR			
		noTW	few	half	many	noTW	few	half	many
TD	w-0	2.8	5.6	6.66	4.43	568	638	757	794
	w-1		3.45	6.14	6.24		645	775	811
	w-2		2.82	6.97	10.56		661	756	794
	w-3		8.14	9.88	9.92		685	711	662
OR	w-0	0.0	10.93	17.42*	28.53	0.0	0.09	0.37	1.41
	w-1		-0.93*	54.96*	18.93*		0.13	0.8	2.55
	w-2		10.42*	15.5*	4.43		0.31	1.89	4.47
	w-3		1.07*	1.34	1.11		0.98	4.66	10.03
FD	w-0	0.0	-2.0	-16.43	-11.0	1.0	0.94	0.74	0.53
	w-1		-4.57	-16.7	-14.9		0.9	0.61	0.38
	w-2		-2.51	-0.57	0.0		0.78	0.41	0.19
	w-3		-1.71	0.0	0.0		0.45	0.1	0.04
CF	w-0	32.55	38.78	43.53	48.14	1.0	1.0	1.0	1.0
	w-1		37.88	43.75	47.67		1.0	1.0	1.0
	w-2		43.1	43.03	44.94		1.0	1.0	1.0
	w-3		41.09	43.87	46.34		1.0	1.0	1.0
DD	w-0	-41.85	-46.66	-49.55	-50.42	1.0	1.0	1.0	1.0
	w-1		-44.37	-49.68	-51.09		1.0	1.01	1.01
	w-2		-45.85	-47.34	-49.25		1.0	1.01	1.02
	w-3		-46.34	-49.49	-49.19		1.0	1.01	1.01

Table 6.9 Time Windows influence. No Balance case.

One of the major limitations of not considering routing decision is the inability to integrate the effect of operational constraints in the design of territories. We have already commented on this in Section 6.4.1. To further demonstrate our point, we ran an additional set of experiments. For each original instance, we consider the territories obtained from the solution of the instance with no time windows. We then use this partition to solve all other versions, with time window, of the same instance. This second phase is done in a day-by-day fashion, similar to Section 6.6.7. In other words, we design the territories without considering time windows, and then we test these on the same instance with time windows. For the sake of brevity we do this only for the No Balance case. This is enough to show how important is to include operational constraints in the algorithm. In Table 6.11, we report the resulting CF and DD. As done for the other sections, we compare TBR and DRR. It is apparent that, regardless of the complexity of the time windows configuration, the territories fail to

		Δ (%)				TBR			
		noTW	few	half	many	noTW	few	half	many
TD	w-0	8.95	19.15	21.9	12.66	737	765	881	879
	w-1		17.14	14.98	12.05		782	868	885
	w-2		20.98	21.39	18.38		795	882	877
	w-3		18.05	19.86	18.09		763	807	747
OR	w-0	-8.0	0.0*	32.5*	75.67*	0.0	0.18	0.49	1.85
	w-1		0.0*	167.1*	20.01	0.0	0.18	1.21	2.67
	w-2		22.73*	23.0*	11.85	0.0	0.39	2.02	4.66
	w-3		6.39*	2.95	1.51	0.0	1.03	4.73	9.98
FD	w-0	2.27	-3.43	-21.14	-17.9	1.0	0.91	0.73	0.52
	w-1		-3.43	-25.84	-13.33		0.91	0.54	0.4
	w-2		-8.91	-1.14	0.0		0.73	0.4	0.19
	w-3		-5.71	0.0	0.0		0.44	0.1	0.04
CF	w-0	48.25	53.29	53.21	52.02	1.0	1.0	0.99	0.99
	w-1		56.36	51.8	48.99		1.0	0.99	0.99
	w-2		55.51	52.24	49.27		1.0	0.98	0.98
	w-3		56.76	50.84	48.29		1.0	0.99	0.99
DD	w-0	-48.85	-54.19	-52.21	-52.58	1.0	1.01	1.03	1.03
	w-1		-54.13	-51.68	-50.02		1.0	1.04	1.02
	w-2		-54.13	-50.39	-48.31		1.01	1.07	1.07
	w-3		-53.41	-51.45	-49.49		1.0	1.03	1.03
RDG	w-0	1.96	18.47	-9.03	38.51	0.18	0.48	0.21	0.19
	w-1		-21.71	1.16	7.89		0.35	0.21	0.22
	w-2		11.75	9.63	-13.15		0.32	0.21	0.19
	w-3		-16.02	5.3	6.15		0.34	0.28	0.22

Table 6.10 Time Windows influence. Duration Case

improve consistency, proving that time windows play a major role in the efficiency of the territories.

In Schneider et al. [2014], the authors conclude that the geographical information is paramount to the design of efficient territories. They do not observe a substantial improvement when integrating time windows related information in the construction phase. However, in the first phase of their algorithm, the sample days are solved with time windows. The solution of the sample days partially defines the pairing costs for the second phase. Therefore, even if not directly, time windows are included in the

territories design.

		Δ (%)			TBR		
		few	half	many	few	half	many
CF	w-0	-0.23	3.26	6.32	0.73	0.74	0.74
	w-1	2.13	6.31	6.99	0.76	0.75	0.73
	w-2	3.76	5.92	5.08	0.75	0.75	0.73
	w-3	0.99	3.92	1.86	0.73	0.73	0.71
DD	w-0	2.7	-2.21	-6.91	2.07	2.06	2.04
	w-1	-1.21	-7.86	-8.2	1.95	1.97	2.0
	w-2	1.67	-4.03	-2.78	2.07	1.98	2.03
	w-3	4.39	-3.39	-0.05	2.1	2.02	2.11

Table 6.11 Time Windows influence. TBR approach using territories designed on the noTW instances.

6.6.11 Demand and Value Variation Influence

This section examines the impact of the variation in demand over the horizon. We proceed as in the previous sections and modify instances of group-A to isolate the factor of interest. Consider one of the original instances, e.g., A-1. For each customer $i \in C$ we denote by $\overline{q}_i$ the average demand over the horizon, computed by considering only the days where i requires service. Consider a parameter $\lambda > 0$. We create a new instance A$^{\lambda}$-1 by setting the demand of a customer $i \in C$ on day $d \in D$ equals to $\max(0, \overline{q}_i + \lambda(q_{id} - \overline{q}_i))$. Therefore, for $\lambda = 0$, if a customer places a request, the demand is always the same. Instead, for $\lambda = 1$, the instance A$^{\lambda}$-1 is the same as A-1. Increasing λ increases the variation in the demand distribution. For each original instance, we create 7 new instances, by varying lambda in the set $\{0, 0.3, 0.6, 1, 1.5, 2, 3, 4\}$. We solve all instances with all cost configuration applying both DRR and TBR approaches. The results are reported in Table 6.12. For each λ, the average is taken over all instances A$^{\lambda}$-a for $a = 1, \ldots, 26$.

Let us first look at the case where no balance requirements are considered.

Clearly, the approach is able to reach an optimal consistency service regardless of the variation. As λ increases, the difference between the two approaches is uniform for all quality measures. It can be observed that increasing the variation decreases the effectiveness of TBR very slightly. However, the effect is contained and not comparable with the effect of other factors considered. In conclusion, if no balance requirements are considered, the variation in the demand of customers does not impact the approach's effectiveness.

	$\lambda =$	0	0.3	0.6	1	1.5	2	3	4
TD	Δ (%)	3.34	3.19	3.12	3.03	3.5	3.54	4.72	6.27
	TBR	588	587	586	587	588	590	600	610
OR	Δ (%)	0.0	0.0	0.0	0.0	0.0*	0.0*	0.0*	2.08*
	TBR	0.01	0.01	0.01	0.01	0.02	0.02	0.03	0.1
FD	Δ (%)	0.0	0.0	0.0	0.0	-0.57	-0.57	-0.57	-1.94
	TBR	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.93
CF	Δ (%)	39.06	42.91	38.83	41.04	37.71	42.22	42.45	43.69
	TBR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-42.46	-45.79	-45.37	-46.0	-44.94	-46.37	-46.64	-48.9
	TBR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 6.12 Demand variation influence. No Balance Case

We now turn our attention to the Duration Case. The results are reported in Table 6.13. Even in this case, the effect of increasing the variation in demand is limited. The same behaviour as the No Balance Case can be observed. The measures vary slightly more due to the fact that introducing the Duration Cost adds complexity to the model and introduces one more cost component to be optimised. Note that, even if Δ-OR and Δ-RDF increase with λ, the actual values of OR and RDG are very low, with RDG being always below the tolerance ρ. Therefore, we can conclude that, even in the Duration Case, variation in demand does not impact the effectiveness of TBR.

Finally, we look at the case with considering the Value Cost. Before presenting the results, we recall that in our data we set value equal to demand, i.e., $q_{id} = v_{id}$. Therefore, when varying the demand of customers and considering the Value Cost, we are looking at the influence that variation in value has on TBR. The results are presented in Table 6.14. For the two highest values of λ, the variation in TD and RVG are not negligible. In these two cases RVG is not under the tolerance threshold anymore and Δ-TD is considerably higher. We can therefore conclude that, when including the Value Cost, a high variation in value impacts the routing quality and balancing ability of TBR. However, we point out that, for $\lambda \in \{3, 4\}$, the variations are quite extreme and not likely to happen in real-world instances.

	$\lambda =$	0	0.3	0.6	1	1.5	2	3	4
TD	Δ (%)	17.66	16.01	19.21	18.85	19.54	20.4	22.63	25.67
	TBR	716	715	726	726	721	737	761	762
OR	Δ (%)	-1.39*	2.08*	2.17*	0.0*	2.08*	-1.39*	1.39*	28.26*
	TBR	0.03	0.02	0.05	0.06	0.03	0.11	0.09	0.15
FD	Δ (%)	-0.71	-1.37	-3.66	-2.29	-1.94	-0.71	-3.09	-3.85
	TBR	0.97	0.98	0.95	0.97	0.97	0.97	0.95	0.9
CF	Δ (%)	52.98	56.79	50.6	54.75	56.69	52.91	53.69	53.79
	TBR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-53.15	-52.62	-52.67	-54.08	-53.83	-53.32	-53.93	-54.01
	TBR	1.0	1.0	1.0	1.0	1.0	1.0	1.01	1.0
RDG	Δ (%)	29.26	71.05	122.3	48.27	4.48	79.05	42.36	51.74
	TBR	0.06	0.07	0.07	0.06	0.05	0.05	0.06	0.06

Table 6.13 Demand variation influence. Duration Case

	$\lambda =$	0	0.3	0.6	1	1.5	2	3	4
TD	Δ (%)	6.12	7.22	7.0	6.77	6.72	10.52	15.68	20.32
	TBR	675	689	681	678	681	699	739	755
OR	Δ (%)	2.08*	0.0*	2.17*	0.0*	0.0*	4.17*	0.0*	0.0*
	TBR	0.02	0.02	0.19	0.02	0.02	0.03	0.03	0.1
FD	Δ (%)	-1.37	-0.57	-5.94	-0.57	-0.57	-2.17	-1.14	-1.71
	TBR	0.98	0.98	0.93	0.98	0.98	0.97	0.97	0.93
CF	Δ (%)	45.09	51.43	45.07	46.69	46.97	48.15	47.94	45.24
	TBR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-48.97	-51.99	-49.62	-50.84	-51.63	-52.73	-51.05	-51.26
	TBR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
RVG	Δ (%)	15.12	33.21	48.07	59.85	131.43	96.23	434.23	244.87
	TBR	0.07	0.07	0.07	0.07	0.1	0.1	0.15	0.17

Table 6.14 Value variation influence. Value Case

6.6.12 Frequency Influence

In this section, we focus on the role played by the frequency of customer demand, i.e. the percentage of days customers require service. In this section, we use instances of group-B. The size of these instances is considerably bigger, with more than a 1000 customers each. For each instance B-b, with $b = 1, \ldots, 4$, we create 10 new instances, denoted by B-b-j, $j = 1, \ldots, 10$. This is done using the following procedure. Fix an instance B-$\bar{b}$. First, we sample a requests' pattern; that is, the set $\{r_d\}_{d \in D}$ where r_d represents the number of requests on day d. In order to mimic the original request patterns, we sample r_d accordingly to a Gaussian distribution $\mathcal{N}(\mu_d, \sigma_d^2)$ with $\mu_d = 100$ and $\sigma_d = 4$ for all days except the last one (which is a Sunday). For the last day we sample μ_d randomly in the interval $[60, 75]$[11]. All 10 new instances B-$\bar{b}$-j have the same requests pattern, i.e. they all have r_d requests on day d. What changes is the set of customers C. For each new instance $j = 1, \ldots, 10$ we sample $j * 100$ customers from the original set. Once we have sampled the set of customers C, on each day d we sample r_d requests from C. Therefore, low values of j mean customers place requests almost every day, while high value of j mean several customers have only one request, and very few have a high number of requests.

For each of the original set of instances we have 10 new instances with decreasing frequency. Note that the size of the newly produced instances is the same, and, for sets of instances created from the same original instance, the geographical distribution cannot radically change. This way we isolate the frequency factor. As we have done so far, we run both DRR and TBR approaches on all instances. We consider all three cases: No balance, Duration and Value. The results are reported, respectively, in Tables (6.15-6.17). In this case, for a fixed j, we average the results over the 4 instances, B-b-j, $b = 1, \ldots, 4$.

Note that, by decreasing the frequency, one has instances with customers placing few requests over the whole horizon. These customer cannot increase CF and DD by much. For this reason, the DRR approach is able to reach a good level of consistency for high values of j. However, for low values of j, that is, instances with customers placing requests almost every day, the TBR approach has a tremendous effect on the consistency. This is true in all three cases: No balance, Duration and Value. Besides this, we observe that frequency does not affect TBR efficiency at all. If balance requirements are considered, the routing quality of TBR is clearly worse but Δ-TD does not change proportionally or inversely to the frequency. Moreover, the unbalance gaps are always below the tolerance value. Summarising, a higher

[11] We choose the intervals for the mean to respect the original proportion of requests for a Sunday, with respect to all other days of the week.

frequency offers more opportunity to improve consistency, but does not impact the efficiency of TBR.

	TD		OR		FD		CF		DD	
j	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR
1	5.13	492	0.0	0.0	0.0	1.0	74.11	1.0	-64.7	1.0
2	5.57	570	0.0	0.0	0.0	1.0	51.76	1.0	-51.33	1.0
3	5.43	569	0.0	0.0	0.0	1.0	38.25	1.0	-41.76	1.0
4	5.53	592	0.0	0.0	0.0	1.0	29.46	1.0	-36.23	1.0
5	5.25	583	0.0	0.0	0.0	1.0	24.53	1.0	-32.31	1.0
6	4.76	591	0.0	0.0	0.0	1.0	17.41	1.0	-25.74	1.0
7	4.25	592	0.0	0.0	0.0	1.0	15.73	1.0	-23.81	1.0
8	5.15	603	0.0	0.0	0.0	1.0	15.04	1.0	-22.98	1.0
9	5.72	618	0.0	0.0	0.0	1.0	12.13	1.0	-19.32	1.0
10	5.1	592	0.0	0.0	0.0	1.0	10.1	1.0	-16.87	1.0

Table 6.15 Frequency influence. No Balance Case.

	TD		OR		FD		CF		DD		RDG	
j	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR
1	25.03	718	0.0	0.0	0.0	1.0	117.22	1.0	-69.99	1.0	-6.1	0.02
2	34.79	881	0.0*	0.14	-7.14	0.93	70.43	1.0	-56.96	1.0	86.74	0.03
3	45.57	961	0.0	0.0	0.0	1.0	43.31	0.98	-44.47	1.04	106.03	0.04
4	39.66	952	0.0*	0.07	-7.14	0.93	32.27	0.99	-37.54	1.02	202.79	0.04
5	44.95	936	0.0*	0.21	-17.86	0.82	24.91	0.99	-32.33	1.02	26.27	0.03
6	36.46	921	0.0*	0.43	-28.57	0.71	20.34	1.0	-28.2	1.01	3.29	0.03
7	42.25	965	0.0*	0.14	-7.14	0.93	16.18	0.99	-23.97	1.01	23.87	0.02
8	38.12	954	0.0*	0.04	-3.57	0.96	14.21	1.0	-21.82	1.01	23.34	0.03
9	36.83	966	0.0*	0.43	-21.43	0.79	12.47	0.99	-19.26	1.03	63.89	0.02
10	41.85	963	0.0*	0.04	-3.57	0.96	8.36	0.97	-13.48	1.06	61.51	0.05

Table 6.16 Frequency influence. Duration Case.

6.6.13 Daily Requests Variation Influence

We now focus on the impact of the variation in the number of daily requests. We modify the instances of group-A as now described. We fix an instance A-a, and denote by r_d the number of requests on day $d \in D$. Moreover, let us denote by $r(\text{A-}a)$ and $\sigma(\text{A-}a)$ the average and standard deviation of the set $\{r_d\}_{d\in D}$. We create 5 new instances, denoted by A-a-v, $v = 1, \ldots, 5$, by modifying the request set $\{r_d\}_{d\in D}$, so that:

- the average number of requests per day does not change, i.e., for all v it holds $r(\text{A-}a\text{-}v) = r(\text{A-}a)$,

	TD		OR		FD		CF		DD		RVG	
j	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR	Δ (%)	TBR
1	8.57	567	0.0	0.0	0.0	1.0	100.73	1.0	-68.39	1.0	117.26	0.04
2	22.51	716	0.0*	0.04	-3.57	0.96	59.82	1.0	-54.95	1.0	45.34	0.03
3	18.0	691	0.0	0.0	0.0	1.0	42.18	1.0	-44.95	1.0	-15.48	0.02
4	15.11	698	0.0	0.0	0.0	1.0	31.02	1.0	-37.43	1.0	10.31	0.03
5	14.85	696	0.0	0.0	0.0	1.0	24.68	1.0	-32.78	1.0	-12.87	0.02
6	20.88	738	0.0*	0.07	-7.14	0.93	18.96	1.0	-27.33	1.0	5.05	0.03
7	17.28	720	0.0	0.0	0.0	1.0	16.4	1.0	-24.59	1.0	37.33	0.03
8	18.09	737	0.0	0.0	0.0	1.0	14.48	1.0	-22.45	1.0	-45.36	0.01
9	19.46	754	0.0*	0.04	-3.57	0.96	13.12	1.0	-20.44	1.0	-11.63	0.02
10	23.07	750	0.0	0.0	0.0	1.0	10.38	1.0	-17.25	1.0	6.09	0.02

Table 6.17 Frequency influence. Value Case.

- the standard deviation is close to some pre-fixed values. In more detail, we have $\sigma(\text{A-}a\text{-}v) \approx \alpha_v * r(\text{A-}a)$, where $\alpha_v = 0, 0.05, 0.1, 0.2, 0.3$, for, respectively, $v = 1, \ldots, 5$.

- The values $\{r_d\}_{d \in D}$ vary (approximately[12]) linearly and the median day has exactly $r(\text{A-}a\text{-}v)$ requests, i.e., since $|D| = 7$, $r_3 = r(\text{A-}a\text{-}v)$.

We solve all of the instances using both DRR and TBR approaches and all three cost configurations: No Balance, Duration and Value cases. We then average the results over all instances A-a-v for $a = 1, \ldots, 26$. The results are reported in Tables 6.18, 6.19 and 6.20

For the No Balance and Duration case the results clearly show that the variation in the number of daily requests has no impact on the efficiency of TBR. As usual TD increases more when balance requirements are considered, but there is also significant fluctuation of Δ-TD which is correlated with the variation of the number of daily requests. In the Value Case the situation is different. Indeed we can observe an increase of Δ-TD proportional to the increase in variation. Moreover, the unbalance gap, RVG, is not always under the tolerance and the gap with DRR is significant. This leads us to conclude that, when the Value Cost is considered, the variation in daily requests has an impact on the efficiency as the balancing of territories become harder.

[12] We use a linear function of d and then round to the nearest integer to determine the values of the r_d's. Therefore, their variation might be not exactly linear. For the same reason, the standard deviation might not equal $\alpha_v * r(\text{A-}a)$, but we choose the linear function so that $\sigma(\text{A-}a\text{-}v)$ is the closest possible to the desired value.

$v =$		1	2	3	4	5
TD	Δ (%)	3.61	3.32	3.49	4.44	4.12
	TBR	609	606	610	611	606
OR	Δ (%)	0.0	0.0	0.0	0.0	0.0
	TBR	0.01	0.01	0.01	0.01	0.01
FD	Δ (%)	0.0	0.0	0.0	0.0	0.0
	TBR	0.99	0.99	0.99	0.99	0.99
CF	Δ (%)	39.58	41.78	46.14	44.92	44.64
	TBR	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-43.41	-43.53	-48.14	-47.23	-46.68
	TBR	1.0	1.0	1.0	1.0	1.0

Table 6.18 Request variation influence. No Balance Case

$v =$		1	2	3	4	5
TD	Δ (%)	26.6	22.53	22.21	22.36	24.11
	TBR	820	777	785	791	791
OR	Δ (%)	4.0*	2.0*	2.0*	2.0*	4.17*
	TBR	0.05	0.02	0.03	0.03	0.07
FD	Δ (%)	-2.09	-1.32	-1.32	-1.32	-4.84
	TBR	0.97	0.98	0.98	0.98	0.95
CF	Δ (%)	65.64	65.2	63.36	65.18	64.88
	TBR	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-55.67	-56.06	-55.45	-54.31	-54.93
	TBR	1.02	1.0	1.0	1.01	1.02
RDG	Δ (%)	102.25	70.41	182.0	89.32	64.93
	TBR	0.08	0.07	0.12	0.06	0.05

Table 6.19 Request variation influence. Duration Case

6.6.14 A Note On Contiguity and Compactness

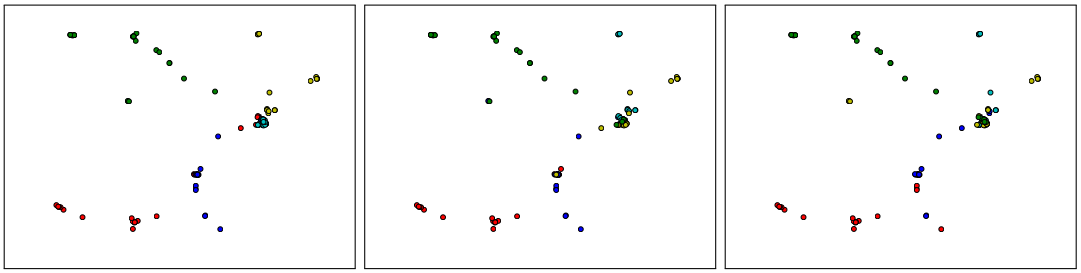
As we previously mention, many authors impose some geometrical requirements when solving a territory design problem. The most two common are contiguity and compactness. Even if there is no formal geometrical definition of these properties, the

$v =$		1	2	3	4	5
TD	Δ (%)	8.79	11.58	10.22	15.06	13.63
	TBR	694	708	715	752	723
OR	Δ (%)	-3.85	-4.0*	-4.0*	-3.85	-4.0*
	TBR	0.02	0.02	0.02	0.02	0.02
FD	Δ (%)	0.64	0.09	0.09	0.64	0.09
	TBR	0.98	0.98	0.98	0.98	0.98
CF	Δ (%)	50.57	54.84	55.78	52.97	55.72
	TBR	1.0	1.0	1.0	1.0	1.0
DD	Δ (%)	-53.87	-52.81	-53.08	-50.65	-52.27
	TBR	1.0	1.0	1.0	1.0	1.0
RVG	Δ (%)	71.8	120.17	160.92	153.26	78.22
	TBR	0.1	0.1	0.12	0.15	0.08

Table 6.20 Request variation influence. Value Case

underlying idea is intuitive. Contiguity means the territories have to be connected, non-nested regions. Compactness means a territory is somewhat round shaped and undistorted. These concepts may vary in the different applications (see Kalcsics et al. [2005] for more details). In the literature, there are several methods used to enforce these properties. Most models try to achieve compactness by adding a p-centre, or p-median, measure in the objective. Although this does not necessarily produce compact territories, the results are usually satisfactory. Moreover, it has the advantage of being computationally tractable. Given that most methods are heuristic, contiguity is usually imposed while constructing a solution. Other approaches involve designing territories of pre-defined shape (Newell and Daganzo [1986]).

In fact, compactness in routing applications may be counter-productive, for example if there are some isolated far customers to be served, just like we have in our problem (see Section 6.6.3). As a side remark, we note that a related concept that could replace compactness is that of *visual appeal* of routes (Constantino et al. [2015]). Using visual appeal has two advantages: it is straightforward to include in our approach and it measures a route property, instead of a territory property. Here we only focus on contiguity. We decided to not enforce contiguity in our method. The ultimate goal is to enhance service consistency and having partial overlap in territories is not necessarily disadvantageous: given that violations of the territories may occur, it might be even more robust to have more than one driver who is



(a) No Balance Cost (b) Duration Cost (c) Value Cost

Figure 6.2: Visual representation of the territories.

familiar with an area (or customer) as is also suggested in Haughton [2008]. In Figure 6.2, for the same instance, we plot the territories associated with three different runs of RBTA, with no Balance Cost, with Duration Cost and with Value Cost. It is possible to appreciate that, when no balance requirement is imposed, the partition (Figure 6.2(a)) is the most visually appealing. When a balance measure is considered (Figure 6.2(b) and 6.2(c)) the partition's contiguity slightly deteriorates. This is necessary, in order to increase the balance. However, even though the areas covered by the territories overlap, these are visually clear and a good degree of contiguity is preserved. Moreover, the areas are similar, with respect to size, to the no balance case. Therefore, we still reach our ultimate objectives of enhanced consistency and increased familiarity of drivers with their area.

6.7 Conclusions and Future Work

In this chapter, we proposed an approach to solve territory design problems. Our method is routing-based. The design of the territories is modelled and solved together with the routing part of the problem. We proposed a simple ALNS-based heuristic that proves to be efficient and able to find good solutions to our problem. We extensively analysed our approach using some real-world inspired instances.

We summarise here our main findings:

- The proposed method is very successful in increasing the consistency of service. This is done without worsening the balance of the routes. Only the routing quality is slightly decreased.
- Increasing the number of territories is not necessarily beneficial if we have also balance requirements. After a certain threshold, this deteriorates the routing quality. However, before the threshold, increasing the number of territories improves the feasibility.

- Balance requirements add considerable complexity to the problem. However, they can be met, without decreasing consistency and routing quality, if some unbalance is tolerated. Conversely, striving for an almost perfect balance and a good consistency level significantly worsens the routing quality of the solution.
- The introduction of time windows in the problems offers more opportunities to improve consistency but affects the routing quality of the solution, although the impact is not as dramatic as might be expected;
- Time windows do not affect the ability of TBR to balance the territories and reach a good consistency level.
- It is important to include the time window constraints in the design of the territories in order to make the approach effective.
- The variation in the demand of customers or in the number of daily requests affect the approach only if the Value Cost is considered and if such variation is high.
- Instances with customers ordering frequently offer more opportunities to improve consistency.
- The territories obtained by the approach are visually appealing. Balance requirements impact the contiguity of the territories, but the magnitude of the effect is low.

It would be interesting to extend our analysis to other operational constraints, such as compatibility constraints or precedence constraints for pick-up and delivery problems, or to other variants of the underlying routing problem, such as the case of heterogeneous fleet. Another potential topic of future work is the sample days selection. Here we assumed the sample days were given, but in reality, one might face a much bigger set of sample days. In those cases, one would have to create a representative subset of days and solve the problem only for those days. However, questions arise on what is the most representative, or the best, subset.

Chapter 6

Conclusions

This thesis investigates several tactical problems that arise in routing applications. Each chapter in Part II focuses on a different problem. During our presentation, we tried to highlight the motivation that led us to consider each of the presented problems. These usually stem from, or are inspired by, real-world problems our research group faced.

Even if the tools used in the solution methods proposed are different, and of a different nature, the methodology of research we followed and the goals we pursued throughout the thesis are similar. As we mention in Chapter 1, our main interests were to extend standard models and solution techniques for VRPs in order to propose new models able to capture the multi-period nature of tactical routing problems and to propose efficient solution methods to such problems; to analyse the impact of the considered approaches at the operational level; to analyse the impact that different features of models and instances have on the efficiency of such approaches.

We believe that the findings presented in this thesis represent a step forward in the area of tactical routing problems. As we already stated, the literature in this area is not quite as developed as that for operational problems. However, tactical problems are of paramount importance for transportation companies, as they influence the daily operations for a medium-to-long horizon. They constitute an area where the use of an optimisation model could lead to significant savings.

The contributions of this thesis were already highlighted in each chapter. Here, we focus on the new questions and possibilities for future work that were generated by the results and findings presented in Part II. We identify and elaborate on new research directions that look worthy of investigation.

Fleet Design with Inter-Day Constraints. In Chapter 4, we pointed out that it could be interesting to introduce inter-day constraints in our approach for fleet design. This is not a straight-forward extension. Indeed, the approach presented in Chapter 4 is based on the ability to treat days independently, using a master problem (in partic-

ular the dual variables of such problem) as the only form of communication. On the other hand, we showed in Chapter 5, how introducing the possibility of splitting the deliveries over consecutive days could lead to significant reduction in the operational costs and fixed size. Combining the two approaches, in a fleet design problem that consider scheduling decisions, could lead to further reduction and augmented efficiency of the fleet, without requiring a complex control policy on the customers' inventory. In particular, the real-world problem presented in Chapter 4 would be suitable for such an extension.

Real-World Application of the Split Deliveries Policy. The work presented in Chapter 5 is at a theoretical level. The chapter served as a first study of the proposed policy and mathematically justifies the approach. However, it would be of interest to implement such approach in a real-world problem. We already elaborated on how the implementation of such a policy should not be hard, and the splitting of some requests may already happen on a non-optimised and non-formalised basis. However, the problem would much more challenging if a rich VRP is considered as the underlying routing problem. Therefore, it is likely that some decomposition methods would be needed. In our view, applying such approach to a real-world problem would constitute a complementary study to the analysis proposed in this thesis.

Territory Design with Heterogeneous Fleet. In Chapter 6, when considering the routing territory design problem, we assumed a fixed homogenous fleet. However, this is not always the case in real problems. While extending our algorithm in order to consider the size of the fleet as a decision variable might not be too hard, considering the case of a heterogeneous fleet might be more challenging. First of all, the problem of balancing the routes would not be as trivial, and one would need to re-define such a concept, for example is different drivers are paid differently. If a fleet design aspect is added on top of the problem, this would fall in the area of fleet design with inter-day constraints, that we mentioned above.

Scenario Selection in Multi-Period Problems. Throughout the thesis we have always assumed the days making up the horizon are given as an input. One point we made, was the fact that considering *all* types of scenarios, and not only a part of them, is important in tactical problems. As an example, in fleet design, considering big days only leads to big fleets, efficient on big days but having a high number of idle vehicles on small days. However, it is not practical to select all possible days. Imagine we have at available several years worth of data. It is clear we cannot con-

sider all of them in our problem. Statistical methods can be used to identify patterns or a set of representative days. However, optimisation methods could be sensitive to the particular set chosen. Moreover, the question of what "representative" exactly means would be raised. It would be of interest to investigate how to choose, or create, a set of days, so to satisfy certain planning criteria. A concrete example is the following: in the territory design problem presented in Chapter 6, one could be more interested in having territories that are very robust to daily changes, or that are not violated, e.g., 70% of the times. Intuitively, the first criterion could lead to the selection of a set of complicating days, which are different between them, so to make the territories as robust as possible. Instead, for the second criteria, it could be better to select of a set of days that well represents the real pattern of requests.

Bibliography

Ahuja, R. K.; Magnanti, T. L.; and Orlin, J. B., 1993. Network flows: theory, algorithms, and applications. (1993).

Amor, H. B. and Desrosiers, J., 2006. A proximal trust-region algorithm for column generation stabilization. *Computers & Operations Research*, 33, 4 (2006), 910–927.

Andersson, H.; Hoff, A.; Christiansen, M.; Hasle, G.; and Løkketangen, A., 2010. Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37, 9 (2010), 1515–1536.

Angelelli, E.; Speranza, M. G.; et al., 2002. The application of a vehicle routing model to a waste-collection problem: two case studies. *Journal of the Operational Research Society*, 53, 9 (2002), 944–952.

Archetti, C.; Bianchessi, N.; and Speranza, M. G., 2011. A column generation approach for the split delivery vehicle routing problem. *Networks*, 58, 4 (2011), 241–254.

Archetti, C.; Bianchessi, N.; and Speranza, M. G., 2015a. A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research*, 64 (2015), 1–10.

Archetti, C.; Campbell, A. M.; and Speranza, M. G., 2014a. Multicommodity vs. single-commodity routing. *Transportation Science*, 50, 2 (2014), 461–472.

Archetti, C.; Jabali, O.; and Speranza, M. G., 2015b. Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, 61 (2015), 122–134.

Archetti, C.; Savelsbergh, M. W.; and Speranza, M. G., 2006a. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science*, 40, 2 (2006), 226–234.

Archetti, C. and Speranza, M. G., 2012. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19, 1-2 (2012), 3–22.

Archetti, C. and Speranza, M. G., 2014. A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2, 4 (2014), 223–246.

Archetti, C.; Speranza, M. G.; and Hertz, A., 2006b. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40, 1 (2006), 64–73.

Archetti, C.; Speranza, M. G.; and Savelsbergh, M. W., 2008. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42, 1 (2008), 22–31.

Archetti, C.; Speranza, M. G.; and Vigo, D., 2014b. Vehicle routing problems with profits. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 273.

Bacao, F.; Lobo, V.; and Painho, M., 2005. Applying genetic algorithms to zone design. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9, 5 (2005), 341–348.

Baldacci, R.; Battarra, M.; and Vigo, D., 2008. Routing a heterogeneous fleet of vehicles. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, 3–27. Springer.

Baldacci, R. and Mingozzi, A., 2009. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120, 2 (2009), 347.

Balinski, M. L. and Quandt, R. E., 1964. On an integer program for a delivery problem. *Operations Research*, 12, 2 (1964), 300–304.

Ball, M. O., 2011. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16, 1 (2011), 21–38.

Bankwest, 2015. Road freight transport industry report. *Connect, Insight for Business*, (2015). https://www.bankwest.com.au/cs/ContentServer?pagename=Foundation/CS/Blob/Document&id=1292539773382&ext=.pdf.

Battarra, M.; Monaci, M.; and Vigo, D., 2009. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36, 11 (2009), 3041–3050.

Beasley, J. E., 1983. Route first—cluster second methods for vehicle routing. *Omega*, 11, 4 (1983), 403–408.

Benzarti, E.; Sahin, E.; and Dallery, Y., 2013. Operations management applied to home care services: Analysis of the districting problem. *Decision Support Systems*, 55, 2 (2013), 587–598.

Bertazzi, L. and Speranza, M. G., 2012. Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1, 4 (2012), 307–326.

Bertoli, F.; Kilby, P.; and Urli, T., 2017a. A column generation-based approach to fleet design problems mixing owned and hired vehicles. *Submitted to International Transactions in Operational Research*, (2017).

Bertoli, F.; Kilby, P.; and Urli, T., 2017b. A general and scalable column generation approach to fleet design problems. *Submitted to Journal of Heuristics, available at arXiv*, (2017). http://arxiv.org/.

Bertoli, F.; Kilby, P.; and Urli, T., 2017c. Vehicle routing problems with deliveries split over days. *Journal on Vehicle Routing Algorithms*, (9 2017), 1–17. doi:10.1007/s41604-017-0002-1.

Bertsimas, D. and Tsitsiklis, J. N., 1997. *Introduction to Linear Optimization*, vol. 6. Athena Scientific Belmont, MA.

Bertsimas, D. J., 1992. A vehicle routing problem with stochastic demand. *Operations Research*, 40, 3 (1992), 574–585.

Boschetti, M. A.; Maniezzo, V.; Roffilli, M.; and Röhler, A. B., 2009. Matheuristics: Optimization, simulation and control. In *International Workshop on Hybrid Metaheuristics*, 171–177. Springer.

Bräysy, O. and Gendreau, M., 2005a. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39, 1 (2005), 104–118.

Bräysy, O. and Gendreau, M., 2005b. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39, 1 (2005), 119–139.

Burke, E. K. and Bykov, Y., 2012. The late acceptance hill-climbing heuristic. *Department of Computing Science and Mathematics University of Stirling—Technical Report CSM-192. ISSN*, (2012), 1460–9673.

Butt, S. E. and Cavalier, T. M., 1994. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21, 1 (1994), 101–111.

Caceres-Cruz, J.; Arias, P.; Guimarans, D.; Riera, D.; and Juan, A. A., 2015. Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47, 2 (2015), 32.

Carter, M. W.; Farvolden, J. M.; Laporte, G.; and Xu, J., 1996. Solving an integrated logistics problem arising in grocery distribution. *INFOR: Information Systems and Operational Research*, 34, 4 (1996), 290–306.

Cattaruzza, D.; Absi, N.; Feillet, D.; and Vidal, T., 2014. A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236, 3 (2014), 833–848.

Ceselli, A.; Righini, G.; and Salani, M., 2009. A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, 43, 1 (2009), 56–69.

Chao, I.-M., 2002. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29, 1 (2002), 33–51.

Chen, Z.-L. and Xu, H., 2006. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40, 1 (2006), 74–88.

Choi, E. and Tcha, D.-W., 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34, 7 (2007), 2080–2095.

Christofides, N.; Mingozzi, A.; and Toth, P., 1981. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11, 2 (1981), 145–164.

Chu, C.-W., 2005. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 165, 3 (2005), 657–667.

Clarke, G. and Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 4 (1964), 568–581.

Coelho, L. C.; Cordeau, J.-F.; and Laporte, G., 2013. Thirty years of inventory routing. *Transportation Science*, 48, 1 (2013), 1–19.

Constantino, M.; Gouveia, L.; Mourão, M. C.; and Nunes, A. C., 2015. The mixed capacitated arc routing problem with non-overlapping routes. *European Journal of Operational Research*, 244, 2 (2015), 445–456.

Cordeau, J.-F. and Groupe d'études et de recherche en analyse des décisions (Montréal, Q., 2000. *The VRP with Time Windows*. Montréal: Groupe d'études et de recherche en analyse des décisions.

Crainic, T. G., 2003. Long-haul freight transportation. In *Handbook of Transportation Science*, 451–516. Springer.

Daganzo, C. F., 1984. The distance traveled to visit n points with a maximum of c stops per vehicle: An analytic model and an application. *Transportation Science*, 18, 4 (1984), 331–350.

Dantzig, G. B. and Ramser, J. H., 1959. The truck dispatching problem. *Management Science*, 6, 1 (1959), 80–91.

Dantzig, G. B. and Wolfe, P., 1960. Decomposition principle for linear programs. *Operations Research*, 8, 1 (1960), 101–111.

Desrosiers, J.; Soumis, F.; and Desrochers, M., 1984. Routing with time windows by column generation. *Networks*, 14, 4 (1984), 545–565.

Doerner, K. F. and Schmid, V., 2010. Survey: Matheuristics for rich vehicle routing problems. *Hybrid Metaheuristics*, 6373 (2010), 206–221.

Dorigo, M.; Birattari, M.; and Stutzle, T., 2006. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1, 4 (2006), 28–39.

Drexl, M., 2012a. Rich vehicle routing in theory and practice. *Logistics Research*, 5, 1-2 (2012), 47–63.

Drexl, M., 2012b. Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46, 3 (2012), 297–316.

Drexl, M., 2013. Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227, 2 (2013), 275–283.

Dror, M.; Laporte, G.; and Trudeau, P., 1994. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50, 3 (1994), 239–254.

Dror, M. and Trudeau, P., 1989. Savings by split delivery routing. *Transportation Science*, 23, 2 (1989), 141–145.

Dror, M. and Trudeau, P., 1990. Split delivery routing. *Naval Research Logistics (NRL)*, 37, 3 (1990), 383–402.

Du Merle, O.; Villeneuve, D.; Desrosiers, J.; and Hansen, P., 1999. Stabilized column generation. *Discrete Mathematics*, 194, 1-3 (1999), 229–237.

Díaz-Madroñero, M.; Peidro, D.; and Mula, J., 2015. A review of tactical optimization models for integrated production and transport routing planning decisions. *Computers & Industrial Engineering*, 88, Supplement C (2015), 518 – 535. doi:https://doi.org/10.1016/j.cie.2015.06.010. "http://www.sciencedirect.com/science/article/pii/S0360835215002697".

Ferland, J. A. and Guénette, G., 1990. Decision support system for the school districting problem. *Operations Research*, 38, 1 (1990), 15–21.

Fisher, M. L. and Jaikumar, R., 1981. A generalized assignment heuristic for vehicle routing. *Networks*, 11, 2 (1981), 109–124.

Francis, P. M.; Smilowitz, K. R.; and Tzur, M., 2008. The period vehicle routing problem and its extensions. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, 73–102. Springer.

Fukasawa, R.; Longo, H.; Lysgaard, J.; de Aragão, M. P.; Reis, M.; Uchoa, E.; and Werneck, R. F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106, 3 (2006), 491–511.

Gaur, V. and Fisher, M. L., 2004. A periodic inventory routing problem at a supermarket chain. *Operations Research*, 52, 6 (2004), 813–822.

Gendreau, M.; Jabali, O.; and Rei, W., 2014. Stochastic vehicle routing problems. *Vehicle Routing: Problems, Methods, and Applications, 2nd edn, Society for Industrial and Applied Mathematics*, (2014), 213–239.

Gendreau, M.; Laporte, G.; and Potvin, J.-Y., 2002. Metaheuristics for the capacitated vrp. In *The Vehicle Routing Problem*, 129–154. SIAM.

Gendreau, M. and Potvin, J.-Y., 2010. *Handbook of Metaheuristics*, vol. 2. Springer.

Gillett, B. E. and Miller, L. R., 1974. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22, 2 (1974), 340–349.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13, 5 (1986), 533–549.

Golden, B.; Assad, A.; Levy, L.; and Gheysens, F., 1984. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11, 1 (1984), 49–66.

Groër, C.; Golden, B.; and Wasil, E., 2009. The consistent vehicle routing problem. *Manufacturing & Service Operations Sanagement*, 11, 4 (2009), 630–643.

Gulczynski, D.; Golden, B.; and Wasil, E., 2010. The split delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review*, 46, 5 (2010), 612–626.

Gurobi Optimization, I., 2015. Gurobi optimizer reference manual. "http://www.gurobi.com".

Hadjiconstantinou, E. and Baldacci, R., 1998. A multi-depot period vehicle routing problem arising in the utilities sector. *Journal of the Operational Research Society*, 49, 12 (1998), 1239–1248.

Hall, R. W.; Du, Y.; and Lin, J., 1994. Use of continuous approximations within discrete algorithms for routing vehicles: Experimental results and interpretation. *Networks*, 24, 1 (1994), 43–56.

Hansen, P. and Mladenović, N., 2014. Variable neighborhood search. In *Search Methodologies*, 313–337. Springer.

Hasle, G. and Kloster, O., 2007. Industrial vehicle routing. *Geometric Modelling, Numerical Simulation, and Optimization*, (2007), 397–435.

Haughton, M. A., 2008. The efficacy of exclusive territory assignments to delivery vehicle drivers. *European Journal of Operational Research*, 184, 1 (2008), 24–38.

Haugland, D.; Ho, S. C.; and Laporte, G., 2007. Designing delivery districts for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 180, 3 (2007), 997–1010.

Hemmelmayr, V.; Doerner, K. F.; Hartl, R. F.; and Savelsbergh, M. W., 2009. Delivery strategies for blood products supplies. *OR Spectrum*, 31, 4 (2009), 707–725.

Hoff, A.; Andersson, H.; Christiansen, M.; Hasle, G.; and Løkketangen, A., 2010. Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research*, 37, 12 (2010), 2041–2061.

Holland, J. H., 1992. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press.

Irnich, S. and Desaulniers, G., 2005. Shortest path problems with resource constraints. In *Column generation*, 33–65. Springer.

Irnich, S.; Schneider, M.; and Vigo, D., 2014a. Four variants of the vehicle routing problem. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 241–271.

Irnich, S.; Toth, P.; and Vigo, D., 2014b. The family of vehicle routing problems. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 1–36.

Jabali, O.; Gendreau, M.; and Laporte, G., 2012. A continuous approximation model for the fleet composition problem. *Transportation Research Part B: Methodological*, 46, 10 (2012), 1591–1606.

Jarrah, A. I. and Bard, J. F., 2012. Large-scale pickup and delivery work area design. *Computers & Operations Research*, 39, 12 (2012), 3102–3118.

Joncour, C.; Michel, S.; Sadykov, R.; Sverdlov, D.; and Vanderbeck, F., 2010. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36 (2010), 695–702.

Jozefowiez, N.; Semet, F.; and Talbi, E.-G., 2007. Target aiming pareto search and its application to the vehicle routing problem with route balancing. *Journal of Heuristics*, 13, 5 (2007), 455–469.

Jozefowiez, N.; Semet, F.; and Talbi, E.-G., 2008. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189, 2 (2008), 293–309.

Kalcsics, J., 2015. Districting problems. In *Location Science*, 595–622. Springer.

Kalcsics, J.; Nickel, S.; and Schröder, M., 2005. Towards a unified territorial design approach—applications, algorithms and gis integration. *Top*, 13, 1 (2005), 1–56.

Karp, R. M., 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, 85–103. Springer.

Kilby, P. and Urli, T., 2016. Fleet design optimisation from historical data using constraint programming and large neighbourhood search. *Constraints*, (2016), 1–20.

Kilby, P. and Verden, A., 2011a. Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In *Proceedings 2nd Workshop on Artificial Intelligence and Logistics (AILOG'11)*, 43–49.

Kilby, P. and Verden, A., 2011b. Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In *Proceedings 2nd Workshop on Artificial Intelligence and Logistics (AILOG-2011)*, 43–48.

Kim, B.-I.; Kim, S.; and Sahoo, S., 2006. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33, 12 (2006), 3624–3642.

Kopfer, H. and Krajewska, M. A., 2007. Approaches for modelling and solving the integrated transportation and forwarding problem. *Produktions-und Logistikmanagement, Springer, Berlin Heidelberg New York*, (2007), 439–458.

Kopfer, H. and Wang, X., 2009. Combining vehicle routing with forwarding: extension of the vehicle routing problem by different types of sub-contraction. *Journal of Korean Institute of Industrial Engineers*, 35, 1 (2009), 1–14.

Kovacs, A. A.; Golden, B. L.; Hartl, R. F.; and Parragh, S. N., 2014. The generalized consistent vehicle routing problem. *Transportation Science*, 49, 4 (2014), 796–816.

Lahyani, R.; Khemakhem, M.; and Semet, F., 2015. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241, 1 (2015), 1–14.

Laporte, G., 2009. Fifty years of vehicle routing. *Transportation Science*, 43, 4 (2009), 408–416.

Laporte, G.; Desrochers, M.; and Nobert, Y., 1984. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14, 1 (1984), 161–172.

Laporte, G.; Mercure, H.; and Nobert, Y., 1986. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16, 1 (1986), 33–46.

Laporte, G.; Ropke, S.; and Vidal, T., 2014. Heuristics for the vehicle routing problem. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 87.

Laporte, G. and Semet, F., 2002. Classical heuristics for the capacitated vrp. In *The Vehicle Routing Problem*, 109–128. SIAM.

Le, T.; Diabat, A.; Richard, J.-P.; and Yih, Y., 2013. A column generation-based heuristic algorithm for an inventory routing problem with perishable goods. *Optimization Letters*, 7, 7 (2013), 1481–1502.

Lee, T.-R. and Ueng, J.-H., 1999. A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*, 29, 10 (1999), 646–657.

Lin, S., 1965. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44, 10 (1965), 2245–2269.

Lourenço, H. R.; Martin, O. C.; and Stützle, T., 2010. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, 363–397. Springer.

Loxton, R.; Lin, Q.; and Teo, K. L., 2012. A stochastic fleet composition problem. *Computers & Operations Research*, 39, 12 (2012), 3177–3184.

Lu, Q. and Dessouky, M. M., 2006. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175, 2 (2006), 672–687.

Lübbecke, M. E. and Desrosiers, J., 2005. Selected topics in column generation. *Operations Research*, 53, 6 (2005), 1007–1023.

M. Gendreau, W. R., O. Jabali, 2014. *Stochastic Vehicle Routing Problems*, chap. 8. MOS-SIAM Series on Optimzation. SIAM.

Madsen, K., 1975. An algorithm for minimax solution of overdetermined systems of non-linear equations. *IMA Journal of Applied Mathematics*, 16, 3 (1975), 321–328.

Maheo, A.; Kilby, P.; and Urli, T., 2016. Fleet size and mix split-delivery vehicle routing. *Submitted.*, (2016).

Malandraki, C. and Daskin, M. S., 1992. Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. *Transportation science*, 26, 3 (1992), 185–200.

Mandal, S. K.; Pacciarelli, D.; Løkketangen, A.; and Hasle, G., 2015. A memetic nsga-ii for the bi-objective mixed capacitated general routing problem. *Journal of Heuristics*, 21, 3 (2015), 359–390.

Maniezzo, V.; Stützle, T.; and Voss, S., 2010. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, vol. 10. Springer.

Marsten, R. E., 1975. *The Use of the Boxstep Method in Discrete Optimization*. Springer.

Matl, P.; Hartl, R.; and Vidal, T., 2017. Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science*, (2017).

Miller, C. E.; Tucker, A. W.; and Zemlin, R. A., 1960. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7, 4 (1960), 326–329.

Mirzaei, S. and Wøhlk, S., 2016. A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, (2016), 1–33.

Mirzaei, S. and Wøhlk, S., 2017. A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, (2017), 1–33.

Moscato, P. and Cotta, C., 2010. A modern introduction to memetic algorithms. In *Handbook of Metaheuristics*, 141–183. Springer.

Mourão, M. C.; Nunes, A. C.; and Prins, C., 2009. Heuristic methods for the sectoring arc routing problem. *European Journal of Operational Research*, 196, 3 (2009), 856–868.

Mourgaya, M. and Vanderbeck, F., 2007. Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, 183, 3 (2007), 1028–1041.

Nagy, G. and Salhi, S., 2007. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177, 2 (2007), 649–672.

Newell, G. F. and Daganzo, C. F., 1986. Design of multiple-vehicle delivery tours—i a ring-radial network. *Transportation Research Part B: Methodological*, 20, 5 (1986), 345–363.

Nikolaev, A. G. and Jacobson, S. H., 2010. Simulated annealing. In *Handbook of Metaheuristics*, 1–39. Springer.

Oppen, J.; Løkketangen, A.; and Desrosiers, J., 2010. Solving a rich vehicle routing and inventory problem using column generation. *Computers & Operations Research*, 37, 7 (2010), 1308–1317.

Ouyang, Y., 2007. Design of vehicle routing zones for large-scale distribution systems. *Transportation Research Part B: Methodological*, 41, 10 (2007), 1079–1093.

Ouyang, Y. and Daganzo, C. F., 2006. Discretization and validation of the continuum approximation scheme for terminal system design. *Transportation Science*, 40, 1 (2006), 89–98.

Parragh, S. N.; Doerner, K. F.; and Hartl, R. F., 2007. A survey on pickup and delivery problems. *Part II: Transportation between pickup and delivery locations, to appear: Journal für Betriebswirtschaft*, (2007).

Parragh, S. N.; Doerner, K. F.; and Hartl, R. F., 2008. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58, 1 (2008), 21–51.

Pecin, D.; Pessoa, A.; Poggi, M.; and Uchoa, E., 2017. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9, 1 (2017), 61–100.

Perrier, N.; Langevin, A.; and Campbell, J. F., 2007. A survey of models and algorithms for winter road maintenance. part iii: Vehicle routing and depot location for spreading. *Computers & Operations Research*, 34, 1 (2007), 211–257.

Pesant, G.; Gendreau, M.; Potvin, J.-Y.; and Rousseau, J.-M., 1999. On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem. *European Journal of Operational Research*, 117, 2 (1999), 253–263.

Pisinger, D. and Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34, 8 (2007), 2403–2435.

Pisinger, D. and Ropke, S., 2010. Large neighborhood search. In *Handbook of Metaheuristics*, 399–419. Springer.

Poggi, M. and Uchoa, E., 2014. New exact algorithms for the capacitated vehicle routing problem. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 59.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31, 12 (2004), 1985–2002.

Prodhon, C. and Prins, C., 2014. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238, 1 (2014), 1–17.

Puchinger, J. and Raidl, G. R., 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 41–53. Springer.

Renaud, J.; Boctor, F. F.; and Laporte, G., 1996a. An improved petal heuristic for the vehicle routeing problem. *Journal of the operational Research Society*, 47, 2 (1996), 329–336.

Renaud, J.; Laporte, G.; and Boctor, F. F., 1996b. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23, 3 (1996), 229–235.

Resende, M. G.; Ribeiro, C. C.; Glover, F.; and Martí, R., 2010. Scatter search and path-relinking: Fundamentals, advances, and applications. In *Handbook of Metaheuristics*, 87–107. Springer.

Ribeiro, R. and Ramalhinho Dias Lourenço, H., 2001. A multi-objective model for a multi-period distribution management problem. (2001).

Ríos-Mercado, R. Z. and Fernández, E., 2009. A reactive grasp for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, 36, 3 (2009), 755–776.

Rodrigue, J.; Comtois, C.; and Slack, B., 2006. *Chapter 7, Concept 3.1—Transport Costs and Rates. The Geography of Transport Systems*. Routledge, New York.

Ropke, S. and Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40, 4 (2006), 455–472.

Rossi, F.; Van Beek, P.; and Walsh, T., 2006. *Handbook of Constraint Programming*. Elsevier.

Rousseau, L.-M.; Gendreau, M.; Pesant, G.; and Focacci, F., 2004. Solving vrptws with constraint programming based column generation. *Annals of Operations Research*, 130, 1-4 (2004), 199–216.

Salazar-Aguilar, M. A.; Ríos-Mercado, R. Z.; and Cabrera-Ríos, M., 2011. New models for commercial territory design. *Networks and Spatial Economics*, 11, 3 (2011), 487–507.

Salhi, S. and Rand, G. K., 1993. Incorporating vehicle routing into the vehicle fleet composition problem. *European Journal of Operational Research*, 66, 3 (1993), 313–330.

Savelsbergh, M. and Song, J.-H., 2007. Inventory routing with continuous moves. *Computers & Operations Research*, 34, 6 (2007), 1744–1763.

Savelsbergh, M. W., 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4, 2 (1992), 146–154.

Schneider, M.; Stenger, A.; Schwahn, F.; and Vigo, D., 2014. Territory-based vehicle routing in the presence of time-window constraints. *Transportation Science*, 49, 4 (2014), 732–751.

Semet, F.; Toth, P.; and Vigo, D., 2014. Classical exact algorithms for the capacitated vehicle routing problem. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 37.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, 417–431. Springer.

Smilowitz, K.; Nowak, M.; and Jiang, T., 2013. Workforce management in periodic delivery operations. *Transportation Science*, 47, 2 (2013), 214–230.

Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35, 2 (1987), 254–265.

Subramanian, A.; Uchoa, E.; and Ochi, L. S., 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40, 10 (2013), 2519–2531.

Sungur, I.; Ren, Y.; Ordóñez, F.; Dessouky, M.; and Zhong, H., 2010. A model and algorithm for the courier delivery problem with uncertainty. *Transportation Science*, 44, 2 (2010), 193–205.

Taillard, É.; Badeau, P.; Gendreau, M.; Guertin, F.; and Potvin, J.-Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31, 2 (1997), 170–186.

Taillard, É. D., 1999. A heuristic column generation method for the heterogeneous fleet vrp. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, 33, 1 (1999), 1–14.

Taillard, É. D.; Gambardella, L. M.; Gendreau, M.; and Potvin, J.-Y., 2001. Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135, 1 (2001), 1–16.

Taillard, É. D.; Laporte, G.; and Gendreau, M., 1996. Vehicle routeing with multiple use of vehicles. *Journal of the Operational Research Society*, (1996), 1065–1070.

Toth, P. and Vigo, D., 2014. *Vehicle Routing: Problems, Methods, and Applications*, vol. 18. Siam.

Tsiligirides, T., 1984. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, (1984), 797–809.

Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Vidal, T.; and Subramanian, A., 2017. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257, 3 (2017), 845–858.

Urli, T. and Kilby, P., 2017. Constraint-based fleet design optimisation for multi-compartment split-delivery rich vehicle routing. In *International Conference on Principles and Practice of Constraint Programming*, 414–430. Springer.

Vanderbeck, F., 1994. *Decomposition and column generation for integer programs*. Ph.D. thesis, Université catholique de Louvain.

Vidal, T.; Crainic, T. G.; Gendreau, M.; and Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234, 3 (2014), 658–673.

Voudouris, C. and Tsang, E., 2003. Guided local search. *Handbook of Metaheuristics*, (2003), 185–218.

Wong, K. and Beasley, J. E., 1984. Vehicle routing using fixed delivery areas. *Omega*, 12, 6 (1984), 591–600.

Wong, R. T., 2008. Vehicle routing for small package delivery and pickup services. *The Vehicle Routing Problem: Latest Advances and New Challenges*, 43 (2008), 475–485.

Yoshizaki, H. T. Y. et al., 2009. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in brazil. *European Journal of Operational Research*, 199, 3 (2009), 750–758.

Zhong, H., 2003. Territory planning and vehicle dispatching with stochastic customers and demand. (2003).

Zhong, H.; Hall, R. W.; and Dessouky, M., 2007. Territory planning and vehicle dispatching with driver learning. *Transportation Science*, 41, 1 (2007), 74–89.

Zoltners, A. A. and Sinha, P., 2005. The 2004 isms practice prize winner—sales territory design: Thirty years of modeling and implementation. *Marketing Science*, 24, 3 (2005), 313–331.