ROBUST OPTIMIZATION OF VEHICLE ROUTING PROBLEMS UNDER UNCERTAINTY

Submitted in partial fulfillment of the requirements for

the degree of

DOCTOR OF PHILOSOPHY

in

Chemical Engineering

ANIRUDH SUBRAMANYAM

B.Tech., Chemical Engineering, Indian Institute of Technology Bombay

Carnegie Mellon University
Pittsburgh, PA

December, 2018

To Amma and Appa

ACKNOWLEDGMENTS

The five years of my PhD have been some of the most influential, productive and fulfilling years of my life. This journey would not have been possible without the constant support and company of several people.

I am deeply indebted to my academic advisor, Prof. Chrysanthos Gounaris, for being an exceptional mentor and role model. Chrysanthos has gone above and beyond his duties as an academic advisor to ensure that I achieve my full potential in whatever I do, be it research, career or otherwise. Chrysanthos, you have taught me how to conduct research. I will always treasure our brainstorming sessions and research meetings where we would figure out an idea or concept together for hours at a stretch, not realizing when time has flown by. Along with the research freedom that you gave me, these interactions always made me feel like we're equals, even in my first few years, and for this I will always be grateful. Your strong work ethic, persistence and attention to detail are qualities that I aspire to in my own professional career. Even outside research, you have actively supported and encouraged me through countless conferences and various teaching and advising opportunities. I feel honored to have been on the receving end of this support and encouragement, and have enjoyed working with you very much.

I would like to acknowledge the members of my doctoral committee–Prof. Ignacio Grossmann, Prof. Lorenz Biegler, Prof. Willem-Jan van Hoeve and Dr. Jose Pinto– for their valuable comments that have led to an improved version of this thesis. I would like to thank Jose in particular for his various inputs and feedback that have significantly increased the practical relevance of this work. I have throughly enjoyed sitting in the lectures of Prof. Grossmann and Prof. Biegler, and will always look up to them as academic role models.

I am grateful for the financial support from the Bertucci Graduate Fellowhip program and the National Science Foundation, under Grants CMMI 1434682 and CBET 1510787.

I would like to thank my collaborators, Prof. Wolfram Wiesemann and Prof. Panagiotis Repoussis, with whom I had many inspiring research discussions. I will cherish the countless skype calls and endless discussions with Wolfram, from whom I have learned a lot about how to succintly present research ideas and concepts. I am also grateful to Panos for his assistance regarding my professional career at various points in my PhD.

During the last five years, I have had the privelege of supervising several talented master's students in their research work. I would like to thank Anushree Kamat, Misna Sameer, Nikhil Apte, Garry Taifan and Tushar Rathi for their hard work, dedication, willingness to pursue open-ended ideas and the several lessons they taught me about how to be an effective mentor. I wish them all the best in their future endeavors.

I feel fortunate to have been part of a very close-knit PSE group and surrounded by several smart people. I would like to thank all members of my research group: Nikos Lappas, Chris Hanselman, Akang Wang and Natalie Isenberg, for all the good times we've had, both in the office and outside it. Akang and Nikos: I will truly miss our discussions on the whiteboard, where several of our ideas have been born. Nikos: thanks for your help with programming, computers and general tech-related stuff, that have made my non-research activities so much easier. A special shout-out to Natalie (and Connor Brem) for all the fun and memorable bike rides!

I am grateful for the several interesting discussions about optimization and math that I've had with: Akang Wang, Devin Griffith, Guru Prashanth, John Eason, Qi Zhang and Saif Kazi. Outside of research, I have some fantastic memories of my time at CMU with– Alex Kazachkov, Braulio Brunaud, Brittany Nordmark, Charles Clerget, Chris Hanselman, Connor Brem, Cristiana Lara, David Bernal, David Thierry, Devin Griffith, Flemming Holtorf, Jake Boes, Javier Lanauze, John Eason, Lisa D'Costa, Markus Drouven, Manzil Zaheer, Natalie Isenberg, Nick Austin, Pablo Garcia-Herreros, Qi Zhang, Ross Cunningham, Saif Kazi, Utsav Awasthi and Wei Wan –thanks to all of you!

This journey has as much been mine as it has of the people who were with me along every step of the way, and with whom I have grown, learned, and immensely enjoyed my time in Pittsburgh. Nidhi Subramanyam and Siddarth Chandrasekharan: thank you for providing me with so many opportunities to travel outside of Pittsburgh. These trips have meant a lot to me over the years. Shweta Sharma and Sundar Venkat: I'm lucky to have been friends with you for as long as I can remember – thanks for all the support and for always being available. Guru Prashanth: you're one of my first friends in Pittsburgh and arguably one of the smartest people I've met during my time here. Thanks for the company, it's been wonderful knowing you. Ankit Gupta and Deepoo Kumar: your apartments have been like homes away from home. Weekends won't be the same without your company. Justin Weinberg: thank you for being a close and loyal friend throughout my time here. I am so glad to have shared some wonderful memories and *classic times* together and I look forward to many more in the future. And finally, Saransh Singh: an attempt to summarize all we've been through would be an exercise in futility. Thanks for everything that I could have asked for in a roommate.

Above all, I would like to thank my parents, Kanchana Subramanyam and Subramanyam Sundareshan. Amma and Appa: none of my experiences would have been possible without your constant support, encouragement and love throughout my academic journey starting from school, to IIT and CMU. You have selflessly supported and believed in me along every step of the way and I hope I can reciprocate this somehow. I dedicate this thesis to you.

ABSTRACT

Vehicle routing problems are a broad class of combinatorial optimization problems that seek to determine the optimal set of routes to be performed by a fleet of vehicles to satisfy given transportation requests. They lie at the heart of transportation operations in supply chains, and constitute one of the most important and intensely studied problems in computational mathematics and operations research. Traditional methods to solve these problems have focused on finding solutions in a deterministic context, in which all input parameters are assumed to be known precisely and in advance, an assumption that is difficult to justify in practical applications. Ignoring uncertainty can lead to solutions that are infeasible or highly suboptimal, and may result in significant economic repercussions when implemented in practice.

The primary goal of this thesis is to develop mathematical tools for the systematic treatment of uncertainty in vehicle routing problems arising at the operational, tactical and strategic levels of planning. A major distinguishing focus of our work is the use of robust optimization, a paradigm for optimization under uncertainty that has received only minor attention in this context. We argue and demonstrate that robust optimization offers a flexible and computationally tractable way to deal with uncertainty in vehicle routing problems. At the operational level, we develop a unified and scalable algorithm to generate vehicle routes that can be feasibly executed when visiting customers with unknown demands. At the tactical level, we study multi-period problems where the goal is to serve customers whose service requests are not entirely known in advance. We introduce a dynamic model which adaptively chooses which requests to serve in each period, as a function of past realizations of the unknown service requests, and develop an algorithm that significantly outperforms traditional methods. At the tactical level, we also contribute the first exact algorithms to design vehicle routes that remain consistent when satisfying variable demands across multiple time periods, and exposit that modest increases in costs can be translated to high levels of service consistency. Finally, at the strategic level, we present a method that enables distributors to postulate generic scenarios of operational uncertainty when allocating long-term delivery time windows to their customers.

The secondary goal of this thesis is to contribute to the broader field of optimization under uncertainty, through the development of theory and algorithms, that are currently lacking but are adequately motivated in vehicle routing applications. In particular, we study dynamic two-stage robust optimization problems with mixed discrete-continuous recourse decisions and present a finite adaptability approximation for their solution. We characterize the geometry of these problems, and present an algorithmic scheme that enjoys strong convergence properties both in theory as well as experiments.

PUBLICATIONS

The following papers related to the work presented in this thesis have been published or submitted in peer reviewed journals:

[1] A. Subramanyam and C. E. Gounaris. "A branch-and-cut framework for the consistent traveling salesman problem." *European Journal of Operational Research* 248.2 (2016), pp. 384–395.

[2] A. Subramanyam and C. E. Gounaris. "A Decomposition Algorithm for the Consistent Traveling Salesman Problem with Vehicle Idling." *Transportation Science* 52.2 (2018), pp. 386–401.

[3] A. Subramanyam, C. E. Gounaris, and W. Wiesemann. "*K*-Adaptability in Two-Stage Mixed-Integer Robust Optimization." *Under Review* (2018). E-print available at https://arxiv.org/abs/1706.07097.

[4] A. Subramanyam, P. P. Repoussis, and C. E. Gounaris. "Robust Optimization of a Broad Class of Heterogeneous Vehicle Routing Problems under Demand Uncertainty." *Under Review* (2018).

[5] A. Subramanyam, A. Wang, and C. E. Gounaris. "A scenario decomposition algorithm for strategic time window assignment vehicle routing problems." *Transportation Research Part B: Methodological* 117 (2018), pp. 296–317.

[6] A. Subramanyam, F. Mufalli, J. M. Laínez-Aguirre, J. M. Pinto, and C. E. Gounaris. "Robust Multi-Period Vehicle Routing under Customer Order Uncertainty." *Under Review* (2018). E-print available at http://www.optimization-online.org/DB_FILE/2017/04/5947.pdf.

The following papers related to the work presented in this thesis have been published or submitted in peer reviewed conference proceedings:

[1] A. Subramanyam and C. E. Gounaris. "Exact Methods for the Multi-Period Traveling Salesman Problem with Arrival Time Consistency." In: *Proceedings of the 6th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2015)*. 2015, pp. 227–230.

[2] A. Subramanyam and C. E. Gounaris. "Strategic Allocation of Time Windows in Vehicle Routing Problems under Uncertainty." In: *Foundations of Computer-Aided Process Operations and Chemical Process Control (FOCAPO/CPC)*. 2017.

[3] A. Subramanyam, C. E. Gounaris, and P. P. Repoussis. "Robust Optimization of Heterogeneous Vehicle Routing Problems under Demand Uncertainty." In: *Proceedings of the 7th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2018)*. 2018, pp. 150–153.

[4] A. Subramanyam, F. Mufalli, J. M. Laínez-Aguirre, J. M. Pinto, and C. E. Gounaris. "Robust Multi-Period Vehicle Routing under Customer Order Uncertainty." In: *Proceedings of the 7th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2018)*. 2018, pp. 154–157.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1

INTRODUCTION

The *vehicle routing problem*, or *VRP* for short, is simple to state:

> Given a set of customers and a fleet of vehicles, determine the cheapest set of vehicle routes to visit all customers using the given fleet. In particular, decide which vehicle visits which customer and in what order so that all routes can be feasibly executed at minimum cost.

The VRP lies at the heart of transportation operations in industrial and commercial supply chains, and it is one of the most important and intensely studied optimization problems in computational mathematics and operations research. However, while there has been significant research done on efficient algorithms for solving the VRP, rapidly changing business models and advances in technology are creating new challenges that are yet unsolved.

In this thesis, we attempt to address the long-standing challenge of dealing with uncertainty in the VRP. Traditionally, the VRP is stated as a deterministic problem, in which all its input data (or parameters) are assumed to be clean, crisp and constant numbers that are perfectly known in advance. Our objective is to relax this assumption and attempt to answer the following question:

> If the parameters needed to describe the VRP are not precisely known at the time when it must be solved, how can we still determine a set of routes that can be feasibly executed under the true (yet, unknown) realization of the parameters?

To that end, we adopt a systems approach, and focus our attention individually to uncertainty-affected routing problems that arise at the operational, tactical and strategic levels of planning. It is natural therefore, that a broad theme of this thesis involves the development of mathematical optimization tools for the systematic treatment of uncertainty. In particular, a major focus of this work is the investigation of *robust optimization* techniques, a paradigm for optimization under uncertainty which has only received minor attention in the context of the VRP, and that too, only in the last five years. As we shall argue and demonstrate in this thesis, robust optimization offers a novel and promising approach to deal with uncertainty in the context of the VRP

because of its simplicity, tractability, and modularity with respect to uncertain parameters. Finally, in the course of our quest to solve the VRP under uncertainty, we also contribute methodologically to the broader field of optimization under uncertainty, through the development of robust optimization theory and algorithms, that are currently lacking but are adequately motivated in our context.

In this introductory chapter, we first motivate and stress the need to study vehicle routing problems under uncertainty in Section 1.1. In Section 1.2, we give a brief overview of the basic components and characteristics of vehicle routing problems. In Section 1.3, we describe the most common vehicle routing parameters that are subject to uncertainty, the applications in which are uncertain as well as an overview of existing approaches for handling uncertainty in vehicle routing problems. We identify several shortcoming of these approaches and the associated challenges in addressing them in Section 1.4. Addressing these challenges is the goal of this thesis for which an outline is presented in Section 1.5.

1.1 THE NEED TO CONSIDER UNCERTAINTY IN VEHICLE ROUTING

The origins of vehicle routing date back to 1959 in the seminal work of Dantzig and Ramser [95], who described a problem of finding the "*...optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal.*" Ironically, the authors proclaimed: "*No practical applications of the method have been made as yet.*" Today, after almost sixty years since the publication of this paper, applications of the VRP are ubiquitous. It spans a wide variety of industries and is routinely solved on a daily basis by thousands of commercial distributors and logistics service providers. Considering that business logistics costs currently account for 7.5% of the United States gross domestic product, and that trucking alone is responsible for transporting 75% of all goods (by value) [167], it is evident that the VRP thus plays an important role in the economic well-being of the nation and the competitiveness, service quality and sustainability of its manufacturing base.

Even beyond commercial distribution, the VRP finds applications in such diverse industries as waste collection, home health care and hospital services, passenger transportation, service technician routing and postal and courier services, among several others [66, 132, 144]. Recently, it is also finding increasing application in the chemical industry, where the goal is to integrate planning and scheduling activities at the plant level with logistics activities at the supply chain level, such as the transportation of inbound raw material. We refer to the textbooks by Toth and Vigo [255, 256], Golden, Raghavan and Wasil [133] as well as the survey [162] for an overview of the wide breadth of applications of the VRP.

Almost all of the aforementioned applications involve routing of vehicles in environments that include significant sources of uncertainty. Examples of such sources are unpredictable traffic conditions, vehicle breakdowns, driver unavailability, missing or im-

precise information on order volumes, and the lack of knowledge of whether a customer (order) will even materialize at all. Section 1.3 provides examples of several industrial applications that are laced with uncertainty. Ignoring the presence of these sources of uncertainty can frequently lead to situations that are either infeasible or suboptimal. For example, if order volumes are uncertain and become known only when customers are visited, a planned route performed by a vehicle may turn out to be infeasible (*i.e.*, may fail to fulfill some orders) if the total observed order volume of the customers scheduled on the route exceeds the carrying capacity of the vehicle. Whenever such a situation occurs, additional costs must be incurred to recover a feasible schedule (for example, the vehicle might have to be dispatched to a central depot to replenish its capacity and/or customers might have to be compensated for poor service). Such situations are highly undesirable, as they result in significant economic and reputational repercussions for distributors, who already operate with fairly low profit margins.

The need for the systematic incorporation of uncertainty has been long recognized by both academics and practitioners. A 2009 commemorative paper by Laporte [176], marking the 50th anniversary of the seminal work of Dantzig and Ramser [95], stresses that "*time has probably come to develop algorithms better able to incorporate dynamic and stochastic features that are so common in practice.*" Yet, a 2014 survey [66] of several commercial vehicle routing software packages found that none of the surveyed VRP tools "utilize fully fledged stochastic VRP methods," with only a few using simulation techniques to assess the robustness of their generated plans. On the one hand, these software tools are being used by thousands of companies to manage their vehicle fleets [66, 152]. On the other, these tools are equipped with modern technologies such as global positioning systems, tracking, telemetry and cloud computing, which collect significant amounts of routing-related data that can be readily analyzed to learn a predictive model of future uncertainty. Evidently, the consideration of uncertainty in the VRP is–and will continue to be–enabled by these modern technological innovations. What remains are modeling and algorithmic innovations that can help realize the huge potential for reducing global transportation costs and positively impacting the society, economy and environment at large. Incorporating uncertainty in the broader context of operations planning and scheduling and mathematical optimization is also of fundamental theoretical importance.

1.2 BASIC COMPONENTS OF VEHICLE ROUTING PROBLEMS

The preamble of this chapter provided a simple definition of the VRP. In practice, however, numerous variants and extensions of the problem exist, reflecting the diversity of operations and constraints encountered in real-world applications. Therefore, the VRP is commonly regarded as a class of problems. From a methodological viewpoint, however, it is useful to focus on a few basic variants that capture the features shared by most vehicle routing problems encountered in practice. The goal of this section is provide a brief overview of this basic, deterministic VRP. We do not provide references

unless absolute necessary, and refer to [133, 255, 256] for a general introduction to basic vehicle routing concepts and terminology.

The VRP is traditionally defined on a *network* (*e.g.*, , road or transportation network) whose nodes represent geographical locations and whose edges represent physical links between these locations. The links have a certain travel cost associated with them. One of the nodes in the network is the depot (or distribution center), in which a fleet of *vehicles* is stationed, while the other nodes represent *customer* locations. The VRP consists of designing *routes*, each to be performed by a single vehicle (refer to Figure 1.1), such that:

(i) each route starts and ends at the depot;

(ii) each customer is visited exactly once on a single route; and,

(iii) the total cost, defined as the sum of the travel costs along the links traveled by the routes, is minimized.

The VRP generalizes the classical traveling salesman problem (TSP) [13], which calls for the determination of a single, minimum cost route visiting all the nodes of the network (*i.e.*, a Hamiltonian cycle), refer to Figure 1.2.

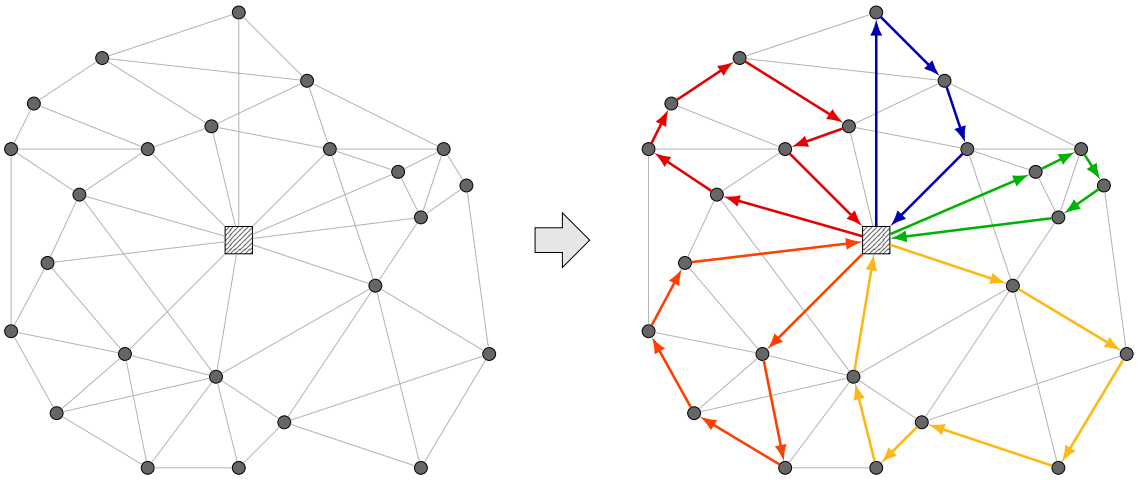


Figure 1.1: Example of a vehicle routing problem (left) and its solution (right). The square represents the *depot*, the circles represent *customers*, the thin gray lines represent the *network* edges while the thick colored lines represent *routes*, each color denoting a route performed by a different *vehicle*.

In the following paragraphs, we describe some common characteristics of the network, vehicles, customers and planning horizon of the VRP, the typical operational constraints (in addition to the above), and the most common objectives to be achieved in the optimization.

NETWORK. The network is generally described through a graph, whose vertices correspond to the depot and the customer locations. Its arcs correspond to the links (e.g.,

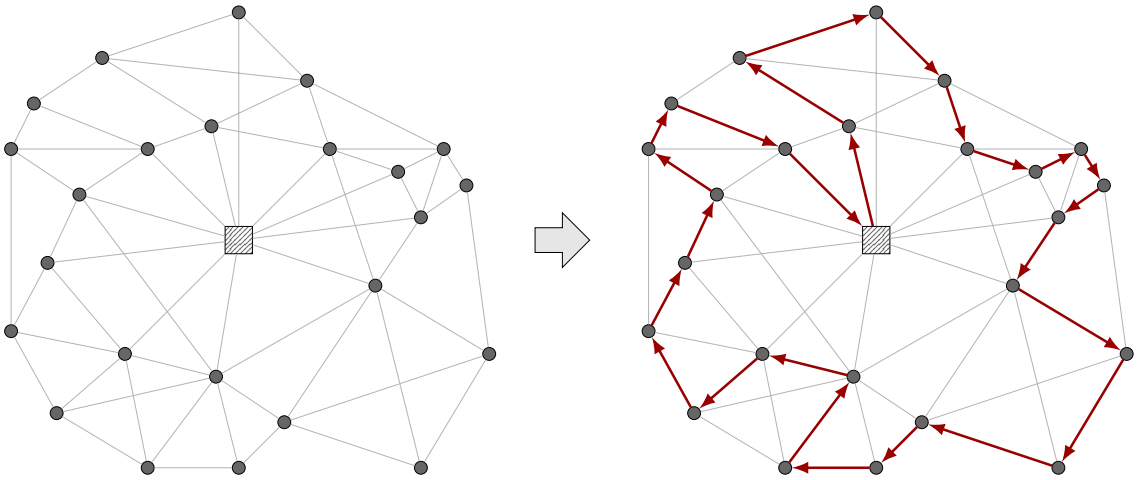


Figure 1.2: Example of a traveling salesman problem (left) and its solution (right).

road sections) between these locations. Each arc is associated with a cost, representing its length, and a travel time, which may be dependent on the vehicle which traverses it, the period (or time of day) during which it is traversed, as well as the direction of travel (the graph being directed in such cases).[1]

VEHICLES. The vehicle fleet represents the primary means to conduct services or transport goods across the transportation network. Typical characteristics of a vehicle include its:

- *cost* (per unit distance, per unit time or per route executed);
- *capacity* (*e.g.*, maximum weight, or volume, or number of pallets);
- *duration limit* (*e.g.*, representing hours of work government regulations);
- subsets of customer locations and arcs of the network which can be traversed by the vehicle (*e.g.*, due to access limitations, size regulations or driver qualifications).

The fleet may be given, or it may be part of the optimization problem, and it is typically heterogeneous (*i.e.*, the vehicles have different characteristics).

CUSTOMERS. The customers place service or transportation requests which require the visit of a vehicle. Typical characteristics of a customer include its

- *location* (*e.g.*, geographical coordinates in a road network);
- *demand* (*e.g.*, amount of goods to be picked up, or delivered);

1 In practice, the actual road network is large but sparse; this network is transformed into a much smaller but dense (often complete) graph. This is achieved by removing all nodes except the depot and customers (*e.g.*, impertinent road junctions) and by defining the inter-customer and depot-to-customer arc costs and travel times to be the weights of the corresponding *shortest paths* in the actual road network.

- *time window* (*e.g.*, days of the week or hours of the day during which the customer is open or its location can be reached due to traffic limitations);
- *service time* (*e.g.*, time required to deliver or pick up goods at the customer location);
- subset of vehicles that can be used to serve the customer (*e.g.*, because of access limitations)

Other characteristics which we do not consider include vendor-managed inventory (where the demand of the customer is not exogenously specified, but controlled by the distributor), split delivery (where the demand can be satisfied by multiple vehicles), and optional requests (where the customer need not be served, but service results in a profit or reward)

PLANNING HORIZON. The planning horizon represents the time period of operation over which routes will be designed and executed. The most common (often implicit) assumption in the VRP is that the planning horizon is of an *operational* nature and spans one period (*e.g.*, one day). In practice, smaller or longer horizons are more appropriate depending on the time period over which routes will be executed.

- Smaller, *real time* planning horizons (*e.g.*, hours or minutes) are considered when routes are dynamically modified during their execution (*e.g.*, because of new or canceled requests, or changed traffic conditions).
- Longer, multi-period planning horizons of a *tactical* nature (*e.g.*, one or more weeks) are common when multiple single-period routes are to be designed. In such cases, the period or day in which a customer request must be served may be fixed *a priori* or determined by the optimization. Moreover, the designed routes may be implemented unchanged in a periodic or cyclic fashion over the course of a longer horizon (*e.g.*, several months) or constantly re-optimized in a rolling horizon fashion over a shorter period (*e.g.*, daily).
- Planning horizons of a *strategic* nature (*e.g.*, several months or years) are more appropriate for decisions such as locating depots, sizing and dimensioning vehicle fleets as well as defining service policies (*e.g.*, allocating time windows to customers).

CONSTRAINTS. The designed routes must satisfy several operational constraints, depending on the characteristics of the customers and the vehicles. The most common constraints are:

- *capacity constraints* of vehicles: the total demand of the customers visited on a route cannot exceed the vehicle capacity;
- *duration limits* of vehicles: the total duration of a route cannot exceed the duration limit of the driver who executes it;

- *time window constraints* of customers: customers can be served only within their time windows on the day (or days) in which they are available;

Apart from these, other examples of constraints include: various forms of *compatibility constraints* (*e.g.*, ensuring that a vehicle can feasibly execute a customer order because of size regulations, or driver qualifications), *precedence constraints* in pickup-and-delivery problems ensuring that a pickup is always performed before the corresponding delivery, and *time-based consistency constraints* in multi-period problems ensuring that visits are performed at the same time (across periods).

OBJECTIVES. The most common objective is to minimize the overall transportation cost, which is often expressed as the sum of fixed costs (*e.g.*, rental, capital amortization or acquisition costs) associated with the used vehicles, and recurring costs (*e.g.*, fuel, labor or insurance costs) that is proportional to the total distance or duration traveled by the vehicles. Other common extensions include minimizing the number of used vehicles, balancing the routes (*e.g.*, with respect to the total duration, or load, or number of customers visited) as well as minimizing penalties associated with violations of the soft constraints (*e.g.*, violation of soft time windows).

ALGORITHMS. The VRP generalizes the TSP and is therefore strongly *NP*-hard [166]. Existing algorithms can be classified as either *exact*, if they compute an optimum solution (or a lower bound on the minimum cost), or *heuristic*, if they cannot establish (sub)optimality of their computed solutions. There exists a vast literature on both exact and heuristic algorithms; see the recent surveys [182, 217, 228]. In the following, we only describe the main ideas behind the most successful algorithms.

EXACT ALGORITHMS: The best exact algorithms are based on *integer programming* [86]. The two most common formulations are the *two-index vehicle flow formulation*, originally proposed in [181], and the *set partitioning formulation*, originally proposed in [33]. The primary decisions in the former are binary variables indicating if an arc of the network is traveled by some vehicle and in the latter are binary variables indicating if a feasible route is used in the final solution. They are both exponential-sized, with the former featuring exponentially many constraints and the latter exponentially many decisions. Therefore, and in contrast to typical mathematical programming formulations, neither of these can be solved with off-the-shelf software. Highly sophisticated custom algorithms, based on *branch-and-cut* and *branch-cut-and-price* have been developed for solving these formulations and they continue to be the subject of active research. We refer to [193] for an efficient implementation of branch-and-cut for solving the two-index formulation and to [30, 210] for an efficient implementation of branch-cut-and-price for solving the set partitioning formulation. These works focus on the *Capacitated VRP* variant in which customers only feature demands and the only routing constraints are the vehicle capacity constraints; however, similar ideas apply for other VRP variants.

HEURISTIC ALGORITHMS: The best heuristic algorithms are all based on *metaheuristics* [60]. The two most common types of metaheuristic algorithms are *local search* methods and *population-based* methods, although the latter intimately rely on the former to achieve good performance. Local search methods start from an initial (not necessarily feasible) *current solution* and at each iteration, move to another solution in some appropriately defined *neighborhood*. The new solution need not be *improving* (in terms of cost or feasibility) and therefore, care must be taken to avoid cycling. The neighborhood definitions are generalizations of their classical TSP counterparts [148, 190] and their size is typically quadratic or cubic (in the number of customers and vehicles). In contrast to local search, population-based methods evolve a population of solutions, stored in some *memory structure*, and then appropriately combine them to generate new solutions that are used to update the memory. We refer to [182] for a succinct survey of metaheuristics for the VRP.

1.3 PARAMETER UNCERTAINTY IN VEHICLE ROUTING PROBLEMS

The aforementioned characteristics of the network, vehicles and customers constitute the input parameters of a VRP. In contrast to the routes, which represent the decisions that need to made to set the operation in motion, these are not subject to selection. In the majority of existing studies, the parameters are assumed to take some constant values before optimization occurs. In practice, however, their true values are subject to significant variability that are often revealed only after the operation takes place. Table 1.1 provides a non-exhaustive list of the parameters associated with the network, vehicles and customers that are typically uncertain, along with some example applications in which they are relevant. It should be noted here that customer requests are different from their demands; the former represents the presence of an order (*e.g.*, for goods or a service), whereas the latter represents the size of the order (typically the amount of goods requested). Similarly, the distinction between time-dependent travel times and uncertain travel times is an important one; the former refers to deterministic, temporal variations resulting from hourly, daily, weekly, or seasonal cycles in traffic volumes, whereas the latter refers to accidents, weather conditions or other random events (e.g., see [195]).

The vast number of applications listed in Table 1.1 have spurred the development of various approaches to deal with parameter uncertainty in vehicle routing problems. A single, unified presentation of these approaches is a difficult task, however, given their significant differences with regard to assumptions as well as terminology. Nevertheless, we attempt to classify the various models as follows: *(i) deterministic reoptimization* (Section 1.3.1), *(ii) Markov decision processes* (Section 1.3.2), *(iii) stochastic programming* (Section 1.3.3), and *(iv) robust optimization* (Section 1.3.4). Of these, the first three have received the widest attention, and there are several surveys and texts dedicated to their review. Therefore, we only highlight the main ideas and provide appropriate references in the relevant sections. The last approach (robust optimization) has received minor

Table 1.1: Examples of uncertain parameters in vehicle routing problems.

Relevant to	Parameter	Example applications
Customers	Demand	vendor-managed inventory systems [71, 85], container repositioning in maritime logistics [91], waste transportation [205], goods distribution [116], design of shared mobility systems [189], crude oil transportation [80], districting and territory planning [147]
	Request	personalized on-demand transportation systems [40, 104], long-haul truckload trucking [219], maintenance scheduling [61, 247], cargo requests in industrial shipping [82, 253], goods distribution [116, 267]
	Service time	local truckload trucking [266], scheduling of service technicians [144], courier delivery [245], orienteering problems [15, 72]
Vehicles	Availability	urban freight distribution [188, 200, 201], dial-a-ride systems [272]
Network	Travel times	sailing times in industrial and tramp shipping [6, 21], disaster management and emergency planning [230, 261], city logistics [141], dial-a-ride cab systems [224]

attention, and that too only in the last few years; therefore, we provide a comprehensive review of the VRP literature that has used this approach.

Before we present the various approaches, it is important to define the *time evolution of information* and associated *extent of decision-making* that is permitted in the application. Specifically, one must establish *(i)* when the uncertain parameters reveal their true values (if at all), and *(ii)* what new routing and assignment decisions can be made (or old ones modified) in response to the revealed information. Often times, these are dictated by the application and the underlying technological, logistical, or contractual infrastructure (which partially explains the diverse assumptions and terminology).

1.3.1 *Deterministic reoptimization*

This approach solves a sequence of deterministic problems at time points that are characterized by the revelation of new information. Specifically, at certain time points (also called *decision epochs*), a deterministic problem is solved to determine the decisions to be executed until the next epoch, and this process is repeated in a rolling horizon fashion. The qualifier "deterministic" refers to the notion that optimization is performed using only the past revealed values (and perhaps, the nominal or expected future values) of the uncertain parameters, without explicitly modeling the uncertainty. For example, if customer requests are unknown and revealed only during the execution of the vehicle

routes, this approach might attempt to determine a 'base routing plan' that services the already received orders and reoptimizes this base plan whenever a new request is received. Variants of this idea include

(i) changing the frequency of reoptimization (*e.g.*, periodically or continuously),

(ii) constraining the optimization using prespecified routing policies (*e.g.*, first-come-first-served) or local update heuristics (*e.g.*, insertion into the existing plan),

(iii) incorporating uncertainty to a limited extent using anticipatory *waiting strategies* (*e.g.*, vehicle waits at the depot for as long as possible) or *multiple plan* approaches (*e.g.*, one amongst multiple solutions is chosen based on a simulation or consensus function).

Deterministic reoptimization approaches are the most widely used methods to deal with parameter uncertainty and are also referred to as (purely) *online* or *dynamic and deterministic* approaches in the vehicle routing literature (*e.g.*, see [35, 160, 184, 216]).

1.3.2 *Markov decision processes*

These approaches attempt to formulate the underlying vehicle routing problem under uncertainty as a Markov decision process (MDP) (*i.e.*, a discrete-time stochastic control process) [41, 220]. Broadly, they can be viewed as *stochastic* generalizations of the deterministic reoptimization approaches described in the previous section, since they incorporate an explicit stochastic model, in which the uncertain parameters are treated as random variables that follow a known probability distribution. These approaches attempt to determine a *policy*, which for each time point (that may be as frequent as the evolution of the uncertain information), defines a function that maps the current *state* of the system (*e.g.*, positions of vehicles, their current load and previous observed realizations of the uncertainty) to *control actions* (*e.g.*, next customer to visit for each vehicle), such that the expected value of the objective function (under the postulated probability distribution) is optimized.

The optimal policy and corresponding optimal *cost-to-go functions* can be characterized using Bellman's equation; however, computing these is computationally intractable in practice, because of the *curse of dimensionality*, and the often large state space. Therefore, several simplifying assumptions are often made to obtain tractable solution algorithms. These include

(i) making simplifying assumptions regarding the probability distribution (*e.g.*, discrete support, independence, sampled scenarios etc.),

(ii) restricting the set of possible decisions (*e.g.*, prevent dynamic assignments of customers to vehicles and only allow dynamic sequencing decisions),

(iii) computing heuristic policies by employing *approximate dynamic programming* (ADP) techniques (*e.g.*, approximating the cost-to-go functions via simulations and per-

forming *approximate policy iteration*, or obtaining the optimal policy over a restricted, often discretized, state space).

MDP-based approaches form a subset of so-called *dynamic and stochastic* approaches for vehicle routing under uncertainty, and they are surveyed in [35, 184, 216, 260].

1.3.3 *Stochastic programming*

Similar to MDP, stochastic programming models the uncertain parameters as random variables that follow a known probability distribution, and seeks to optimize a risk measure (such as the expected value, or *conditional-value-at-risk*) of some cost function [58, 229]. The two basic modeling approaches in this category are two-stage recourse models and chance-constrained models.

1. *Two-stage recourse models.*
   This approach consists of modeling the problem in two stages. In the first stage, a base plan, or *a priori* or *here-and-now* solution is designed, before the realizations of the uncertain parameters are known. In the second stage, the first stage solution is executed and *recourse* actions or *wait-and-see* decisions based on a predetermined policy are taken in response to the revealed uncertain information.

   For example, suppose that customer demands are uncertain and revealed only upon visiting the customer locations. In a typical two-stage recourse model, the planned solution corresponds to vehicle routes that are chosen before the uncertain demands are revealed. Since it is possible that a vehicle may reach a customer and not have sufficient capacity to serve its realized demand (referred to as a 'route failure'), recourse actions must be taken to ensure feasibility. A classical recourse policy is to return to the depot upon failure, offload, and resume collections by continuing along the planned route from the point of failure.

   The two-stage recourse model may be viewed as a restriction of an MDP in which the optimal policy is restricted to the class of predetermined recourse policies. For example, a typical MDP model of stochastic demands imposes no planned solution, and allows the vehicle to visit any location at any point of time (including the depot even if there is no failure). Therefore, the MDP models typically give better objective values. The tradeoff is that two-stage recourse models are computationally more tractable than MDP models. Nevertheless, they are also affected by the curse of dimensionality and tractable algorithms make several simplifying assumptions regarding:

   (i) the probability distribution (*e.g.*, Gaussianity, discrete support, independence),

   (ii) the operation (*e.g.*, no more than a certain number of failures per route), or

   (iii) the recourse policies (*e.g.*, the recourse 'action' is to simply evaluate the sum of incurred penalties of constraint violation)

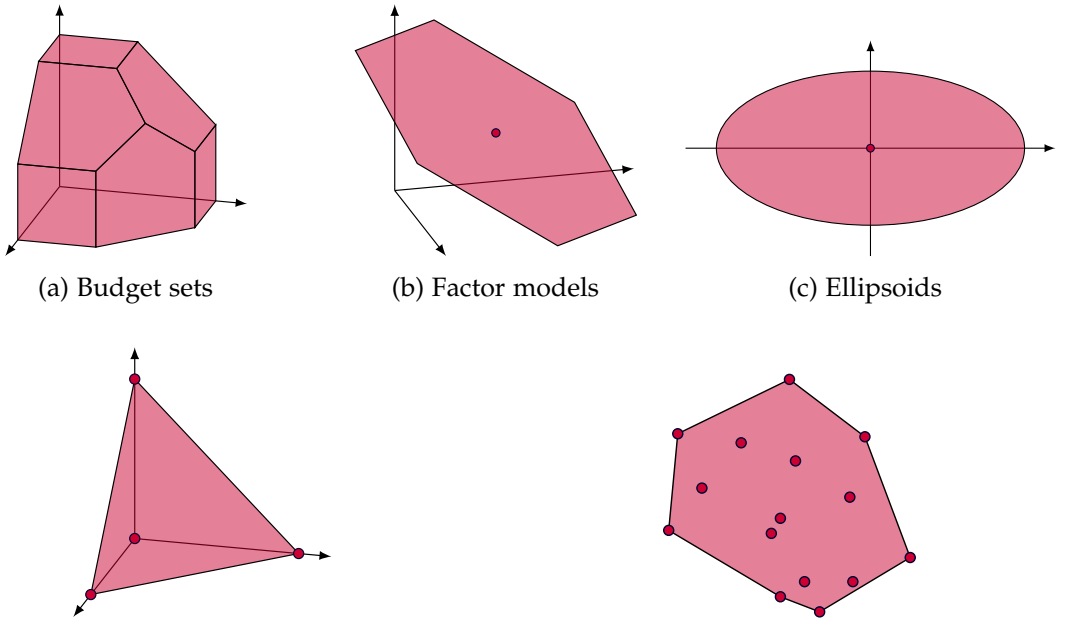
The models are typically formulated using extensions of the two-index vehicle flow or set partitioning formulations (see Section 1.2) and solved using stochastic programming based algorithms such as Benders' or Lagrangean decomposition (e.g., see [177]).

2. *Chance-constrained models.*
   A traditional chance-constrained approach models the problem in a single stage; the goal is to select a base plan, or *a priori* solution that satisfies all constraints with a prespecified probability. In contrast to two-stage recourse models, in which route failures are allowed and 'repaired' by taking recourse actions, chance-constrained models assume that failures are not allowed (*e.g.*, because drivers take the same route every day on a routine basis), and seeks to limit its probability. However, similar to two-stage recourse models, computation of this probability becomes rapidly intractable for arbitrary distributions. Therefore, similar simplifying assumptions (*e.g.*, independent and identically distributed parameters, or Gaussianity) must be made to obtain tractable solution algorithms.

The literature on stochastic programming approaches for vehicle routing problems is surveyed in [73, 125].

1.3.4 *Robust optimization*

Compared to dynamic and stochastic programming, robust optimization is a more recent approach for optimization under uncertainty, in which the model of parameter uncertainty is not a probabilistic one, but rather deterministic and set-based. The goal is to determine decisions that remain feasible for *any* realization of the uncertain parameters in a given, prespecified *uncertainty set*. The motivation for this set-based approach is two-fold. First and foremost, it overcomes the curse of dimensionality that plagues traditional approaches and results in an optimization problem that is often as computationally tractable as the original deterministic problem. Second, this approach may be the only reasonable alternative if distributional information is not readily available (*e.g.*, if there is insufficient data to estimate distributions). Even if sufficient data is available, the tractability benefits may make robust optimization more attractive than existing dynamic and stochastic programming approaches. In such cases, it is possible to choose the structure and size of the uncertainty set to provide *a priori* probabilistic guarantees on the performance of the robust solution (similar to chance-constrained models in stochastic programming), as well as incorporate additional information about the distribution (*e.g.*, support, symmetry and moments) in the uncertainty set (which has led to the field of *distributionally robust optimization*). Some examples of popular and well-motivated classes of uncertainty sets are shown in Figure 1.3 (a precise definition of these sets is provided in Chapter 2). We refer to [37, 42, 135] for an overview of the theory of robust optimization.



(a) Budget sets (b) Factor models (c) Ellipsoids

(d) Cardinality-constrained (Gamma) sets (e) Discrete sets

Figure 1.3: Examples of uncertainty sets; each dimension corresponds to an uncertain parameter.

Robust optimization has only recently started receiving attention in the context of vehicle routing problems. Table 1.2 provides a comprehensive overview of papers that have applied robust optimization to address parameter uncertainty in the VRP. Almost all of the listed papers strive toward the goal of determining a set of vehicle routes that satisfy their constraints (capacities in case of demand uncertainty, and time windows and duration limits in case of service and travel time uncertainty) for all possible realizations of the uncertain parameters from the considered uncertainty sets. The major differences lie in their assumptions regarding the structure of the uncertainty sets and the corresponding solution algorithms, which are often intimately tied to the structure of the set. The majority of the papers assume that the uncertainty set is the *cardinality-constrained* or *gamma* set, originally proposed in [57], and rely on algorithms that are based on modifications or extensions of classical formulations of the deterministic VRP (see Section 1.2).

All of the papers listed in Table 1.2 have focused on finding *static* solutions that cannot adapt to observed values of the uncertain information. Allowing solutions to *dynamically* adapt to the observed uncertainties will result in lower costs, whenever such adaptations are technologically feasible. One reason for this shortcoming is that the early literature on robust optimization theory models the problem in a single stage, and determines a *here-and-now* solution that remains feasible for any realization of the parameters in the chosen uncertainty set. Recent theoretical contributions allow the consideration of dynamic two-stage and multi-stage models, in which *recourse* actions can be taken in view of the revealed uncertain information, similar to recourse models in stochastic

Table 1.2: Overview of robust optimization approaches to vehicle routing under uncertainty.

Reference	Uncertainty in	Uncertainty set	Key features
[234–237]	Travel costs	Discrete	various metaheuristics
[245]	Service time	Discrete	insertion heuristic
[78]	Service time	Gamma	arc routing, branch-and-cut
[207, 230]	Travel times	Interval	reduces to deterministic VRP
[5, 6, 63, 254]	Travel times	Gamma	path inequalities, branch-and-cut algorithm, ant colony search, large neighborhood search
[244]	Demand	Interval	reduces to deterministic VRP
[109]	Demand	Interval	detours-to-depot modeled using stochastic programming, tabu search
[75, 243]	Demand	Gamma	open routes, minimize infeasibility penalties
[215]	Demand	Gamma, Budgets	branch-price-and-cut algorithm
[137, 138]	Demand	Polyhedra, Budgets, Factor models	connection to chance-constraints, robust counterparts of classical formulations, branch-and-cut algorithm, adaptive memory programming
[7, 55]	Demand + Travel times	Gamma	inventory routing, problem-specific decomposition and heuristics
[153, 185]	Demand + Travel times	Gamma	branch-and-price algorithm, variable neighborhood search
[202, 264]	Demand + Travel times	Gamma	uncertainty set-specific formulation, branch-price-and-cut algorithm, insertion heuristic
[2, 159, 278]	Demand + Travel times	Moment ambiguity set	distributionally robust optimization, new risk measure, Benders' algorithm

programming and Markov decision processes. However, these contributions are relatively recent and multi-stage robust optimization is an active area of research. We defer the literature review to Chapter 6. For now, we simply note that most contributions have focused on models in which the recourse actions are continuous decisions, while the literature on models in which the recourse actions include discrete decisions is relatively sparse. It is these models that are most relevant to this thesis, as recourse actions in vehicle routing problems are assignment and sequencing decisions, that are inherently combinatorial and discrete-valued.

1.4 CHALLENGES AND LIMITATIONS IN EXISTING APPROACHES

In view of the shortcomings of existing approaches, we have identified several key challenges in the modeling and solution of vehicle routing problems under uncertainty. In fact, we have also identified gaps in general robust optimization methodology which bear relevance not only to vehicle routing, but to a much wider class of mathematical optimization problems. Specifically, we identify the following challenges that we would like to address in this thesis.

1. *Dynamic robust optimization with discrete recourse decisions.*
   Two-stage robust optimization problems with discrete recourse actions and their multi-stage variants are not fully understood and we lack efficient schemes for their solution. A key challenge is to make some progress towards gaining a better theoretical understanding of their geometry, as well as design efficient algorithms with provable guarantees (at least for the two-stage version).

2. *Dynamic robust optimization in vehicle routing problems.*
   All robust optimization approaches to vehicle routing problems have focused on finding *static* solutions of single-stage models, in which no (discrete) recourse actions can be taken in response to the revelation of uncertain information, preventing their applicability to problems in which routing decisions can be *dynamically* adapted in a multi-stage fashion. Current approaches to such uncertainty-affected routing problems are based on Markov decision processes and stochastic programming, that are plagued by the curse of dimensionality, and are hence intractable. A key challenge is to extend robust optimization to (at least) two-stage vehicle routing models in a way that retains its computational tractability.

3. *Uncertainty in customer requests.*
   Customer requests represent one of the most common sources of uncertainty in practical applications (see Table 1.1). Yet, none of the existing robust optimization approaches have attempted to address this uncertainty (see Table 1.2). This is primarily because of the difficulties in representing and constructing an uncertainty set of discrete-valued uncertain parameters. In contrast to demand and travel times, that are continuous-valued and allow the reformulation of the corresponding robust optimization problem to a finite-dimensional deterministic model, the presence or absence of a customer order is a discrete event, providing more challenges for robust optimization modeling and solution algorithms.

4. *Unified, scalable methods for static robust routing.*
   It is becoming increasingly evident that designing *a priori* or *static* routes is most tractably achieved via robust optimization as opposed to stochastic or chance-constrained programming (at least for demand uncertainty). Yet, there is an evident lack of a unified, flexible method that has the potential of being applied to a wide range of problem settings. Specifically, the challenge is to design algorithms for static robust routing that are modular with respect to *(i)* the choice of the uncertainty

set, *(ii)* the solution algorithm (particularly in the case of metaheuristics), and to an extent *(iii)* any side constraints in the underlying vehicle routing problem. In keeping with the robust optimization paradigm, the challenge is also to ensure that the method can scale to instances of practical size.

5. *Explicit consideration of service consistency.*
   One of the primary drivers for research in static or *a priori* routing are its important practical benefits. Notably, it results in regular or consistent routes that assign the same driver to the same set of customers to serve them at roughly the same time. Such consistent routes are easy to adapt to the realization of the daily uncertainties. Moreover, this consistency help companies realize the important goal of personalization of services while also improving driver productivity and familiarity with their daily routes and territories. A key challenge is to explicitly incorporate service consistency in solution algorithms for vehicle routing problems as a means to hedge against variable demand and service times.

6. *Strategic models in vehicle routing.*
   Strategic models in vehicle routing problems have not received a lot of attention, with most papers focusing on traditional facility (*e.g.*, depot) location and fleet dimensioning models. Recently, there has been some work done toward strategically allocating customers' time windows, that are inherent to virtually all routing problems. One challenge is that operational level information (such as customer demand or travel times) is often not known with certainty at the strategic level, and only scenario-based models are available to represent this uncertainty. The challenge is to develop efficient, scalable, algorithms that can utilize the vast arsenal of solvers for the deterministic VRP in an oracle fashion and that can scale to virtually any number of postulated scenarios.

1.5 AIMS AND OUTLINE OF THE THESIS

The aim of this thesis is to develop mathematical tools for the optimization of vehicle routing problems under uncertainty. To that end, we adopt a systems approach and aim to address each of the six challenges highlighted in the previous section at the operational, tactical and strategic levels of planning (see Table 1.3). A strong focus of our work is on developing optimization algorithms that not only have strong theoretical guarantees but are also simple to implement in practice. The aim is to enable the reinvention of many algorithms that are proven to work well for deterministic vehicle routing problems in a new capacity as algorithms for vehicle routing under uncertainty. Throughout the thesis, we highlight the effectiveness of our proposed schemes via extensive experiments on standard test cases from various application domains, and whenever applicable, compare the performance of our algorithms as well as the obtained solutions against those of other methods. In the following, we provide an overview of the remaining chapters of the thesis.

Table 1.3: Overview of chapters 2–6, listing which of the challenges are addressed in each chapter.

#	Chapter title	Challenges addressed
2	*Operational* routing under demand uncertainty	Unified, scalable methods for static robust routing
3	*Tactical* planning under customer order uncertainty	Dynamic robust optimization with discrete recourse actions, Dynamic robust optimization in vehicle routing, Uncertainty in customer requests
4	*Tactical* enforcement of service consistency	Explicit consideration of service consistency
5	*Strategic* allocation of time windows	Strategic models in vehicle routing
6	*K*-adaptability in two-stage robust optimization	Dynamic robust optimization with discrete recourse actions

In Chapter 2, we aim to develop a unified, scalable method to determine vehicle routes that remain robust during their execution. We consider an extended model of the classical VRP, where a mixed fleet of vehicles with different capacities, fixed and routing costs is used to serve customers. This model includes as special cases, all variants of the so-called *heteregeneous* and *fleet size and mix* problems studied in the literature, as well as problems with general *site dependencies* and *multiple depots*. We consider uncertainty in customer demands, and seek to determine a minimum-cost set of vehicle routes that remain feasible for all anticipated demand realizations from a given uncertainty set. To solve this problem, we develop robust versions of classical *local search* moves, that are at the heart of all metaheuristic approaches (see Section 1.2), and establish that efficient evaluation of the local moves under demand uncertainty can be achieved for five broad classes of uncertainty sets. The proposed local search is shown to be modular by incorporating it into two standard metaheuristics to determine robust solutions. The quality of the metaheuristic solutions is quantified using a novel integer programming model that provides lower bounds on the optimal solution. An extensive computational study on literature benchmarks shows that the proposed method allow us to obtain robust routes with minor additional effort compared to deterministic routes and that they are of high quality, in general.

In Chapter 3, we study multi-period vehicle routing problems where the aim is to determine a visit schedule and associated routing plan for each period using capacity-constrained vehicles. We consider uncertainty in customer service requests, and allow customers to place their requests dynamically over the planning horizon. To guarantee the generation of routing plans that can flexibly accommodate potential requests that have not yet been placed, we model future potential customer requests as binary random variables, and we seek to determine a visit schedule that remains feasible for all anticipated realizations of the requests. To that end, the planning process can be viewed as a multi-stage robust optimization problem with binary recourse decisions. We approximate the multi-stage problem via a non-anticipative two-stage model for which we propose a

novel integer programming formulation and a branch-and-cut solution approach. To investigate the quality of the solutions we obtain, we also derive a valid lower bound on the multi-stage problem and present numerical schemes for its computation. Monte Carlo simulations on a rolling horizon show that our approach is practically tractable and generates high quality robust plans that significantly outperform existing approaches in terms of both operational costs and fleet utilization.

In Chapter 4, we aim to explicitly incorporate service consistency constraints over a tactical planning horizon. The goal is to identify the minimum-cost set of routes that a single vehicle should follow during the multiple time periods of a planning horizon, in order to provide consistent service to customers. The requirement for consistent service corresponds to restricting the difference between the earliest and latest vehicle arrival-times, across the multiple periods, to not exceed some given allowable limit. We present two exact algorithms for this problem. The first is a branch-and-cut algorithm that can solve three novel mixed-integer linear programming formulations of the problem. The second is based on a decomposition of the problem into a sequence of classical traveling salesman problems with time windows. Extensive computational experiments show that the decomposition algorithm is better, as it can solve instances whose sizes are representative–or even exceed–expected sizes of real-world distribution settings involving a single vehicle. We empirically show that *(i)* arrival-time consistency can be achieved with merely a small increase in total routing costs, and *(ii)* the cost of implementing consistent routes can be reduced significantly if vehicles are allowed to idle en route.

In Chapter 5, we study the strategic planning problem of assigning time windows to customers. This problem can be viewed as a two-stage stochastic programming problem, where time window assignments constitute first-stage decisions, vehicle routes adhering to the assigned time windows constitute second-stage decisions, and the objective is to minimize the expected routing costs. We develop a new scenario decomposition algorithm to solve the sampled deterministic equivalent of this stochastic model. From a modeling viewpoint, our proposed approach can accommodate general time window structures as well as general scenario-based models of uncertainty for several routing-specific parameters, including customer demands and travel times, among others. From an algorithmic viewpoint, our approach is easily parallelized, can utilize any deterministic VRP solver as a black box, and can be easily modified as a heuristic for large-scale instances. A comprehensive computational study demonstrates that our algorithm strongly outperforms all existing solution methods, and quantifies the trade-off between computational tractability and expected cost savings when considering a larger number of scenarios during strategic decision-making.

In Chapter 6, we study two-stage robust optimization problems with mixed discrete-continuous recourse decisions in both stages. Despite their broad range of applications, these problems have posed two fundamental challenges: *(i)* they constitute infinite-dimensional problems that require a finite-dimensional approximation, and *(ii)* the presence of discrete recourse decisions prohibits duality-based solution schemes. We

address the first challenge by studying a K-adaptability formulation that selects K candidate recourse policies *before* observing the realization of the uncertain parameters and that implements the best of these policies *after* the realization is known. We establish conditions under which the K-adaptability problem remains continuous, convex and tractable, and we contrast them to the corresponding conditions for the two-stage robust optimization problem. We address the second challenge through a branch-and-bound scheme that enjoys asymptotic convergence in general and finite convergence under specific conditions. We illustrate the performance of our algorithm in numerical experiments involving benchmark data from several application domains.

To aid readability, the nomenclature used in each chapter is summarized in an appendix provided at the end of the chapter.

2

OPERATIONAL ROUTING UNDER DEMAND UNCERTAINTY

In this chapter, we aim to develop a method that can tackle vehicle routing problems in which the vehicles execute their routes over an operational horizon (*e.g.*, one day), but customer demands are unknown prior to dispatch. To deal with this uncertainty, we shall seek to determine a minimum-cost set of routes that are *robust*; that is, the routes must be such that the total demand of the customers served on any route must never exceed the vehicle capacity for any realization of the customer demands from a given *uncertainty set*.

To solve this problem, we develop robust versions of classical *local search* moves. The evaluation of the local moves amounts to the solution of a convex optimization problem in general but we establish that it can be done much more efficiently (in fact, in closed form) for several broad classes of uncertainty sets. We illustrate the modularity of the proposed local search by incorporating it into two standard metaheuristic algorithms.

The qualities of the metaheuristic solutions are quantified using a new integer programming formulation that provides lower bounds on the optimal solution. Interestingly, we show that efficient solution of this formulation via a branch-and-cut algorithm is enabled by the same principles that underly the efficient evaluation of local moves in a heuristic algorithm.

Finally, in addition to the algorithm, we show that our method is also modular in terms of problem features. In particular, we study an extended model of the classical VRP, that includes as special cases, all variants of the so-called *heteregeneous* and *fleet size and mix* problems studied in the literature, as well as problems with general *site dependencies* and *multiple depots*. An extensive computational study on literature benchmarks shows that the proposed method is also scalable across instances of these variants.

The rest of this chapter is organized as follows. After further motivating and providing some background in Section 2.1, Section 2.2 provides a mathematical definition of the various VRP variants that we consider in this paper. Section 2.3 presents the various classes of uncertainty sets, the closed-form expressions of the local moves as well as data structures for their efficient computation. Section 2.4 presents the robust local search and its incorporation into metaheuristics. Section 2.5 presents the integer programming

formulation and branch-and-cut algorithm. Section 2.6 presents computational results; and, Section 2.7 presents a summary of the key results from this chapter.

2.1 MOTIVATION AND BACKGROUND

Vehicle routing problems involve the determination of cost-optimal transportation plans for the distribution of goods and/or services between production facilities, distribution centers and end customers. Despite their differences, a common goal of all vehicle routing problems is to determine an optimal assignment of customer orders to vehicles, as well as the optimal sequencing of customer orders served by individual vehicles. The most common objective is to minimize the transportation cost, which is often expressed as the sum of one time costs (e.g., rental or capital amortization costs of individual vehicles) that is proportional to the size of the vehicle fleet, and/or recurring costs (e.g., fuel, labor or insurance costs) that is proportional to the total distance or duration traveled by individual vehicles.

The most widely studied variant of the vehicle routing problem is the Capacitated Vehicle Routing Problem (CVRP) [176]. The CVRP aims to determine the optimal delivery of goods from a depot to a set of customers using capacity-constrained vehicles. Traditionally, the literature on the CVRP makes two simplifying assumptions: *(i)* the vehicle fleet is assumed to be homogeneous, fixed and stationed at a single depot, and *(ii)* the problem data, such as customer demands and transportation costs, are assumed to be precisely known when the problem is to be solved.

The aforementioned assumptions are often difficult to justify in practice. First, the vehicle fleet is rarely homogeneous in most practical applications. We refer the reader to [151] who provide an excellent overview of practical aspects of fleet sizing and dimensioning that arise in real industrial applications. The authors argue that a vehicle fleet that is acquired over a long period of time is often heterogeneous not only because the acquired vehicles inherently have different physical characteristics, but also because they develop different characteristics over their lifetimes (e.g., operating, maintenance and insurance costs vary depending on the level of depreciation and usage). Moreover, distributors typically want a diverse vehicle fleet, both due to operational constraints (e.g., physical dimensions or compatibility constraints that restrict access of certain vehicles to certain areas) as well as the inherent benefits of owning a versatile fleet.

Second, the assumption of deterministic problem data is unrealistic. Indeed, the parameters of a vehicle routing problem are often subject to significant uncertainty, and their precise values are often only observed gradually during the execution of the transportation plan. For example, travel and service times can vary due to unforeseen events such as bad weather, mechanical breakdowns or traffic congestion. Similarly, customer demands fluctuate from day to day and in fact, they may be uncertain even at the time when the vehicles are to be dispatched. The motivation for taking into account this uncertainty is particularly strong when making strategic or tactical fleet composition

decisions. This is because, on the one hand, operational parameters such as customer demands and travel times are often not known with certainty at the strategic level when the fleet composition is to be decided. On the other hand, these long-term decisions often involve significant amounts of capital investment; therefore, when customer demand is higher than expected, external vehicle fleets must be leased over a short-term operational horizon, which is also costly.

The goal of this chapter is to lift the aforementioned assumptions of fixed, homogeneous fleets and deterministic parameters in vehicle routing problems. In particular, we study a generalization of the CVRP, known as the Heterogeneous Vehicle Routing Problem (HVRP), and focus our attention to problem settings where the customer demands are subject to uncertainty. The HVRP model is very powerful, since it subsumes a number of other problem variants, including but not limited to, the Fleet Size and Mix and Multi-Depot Vehicle Routing Problems. The HVRP model was introduced in the seminal work of [134], and since then, several papers have studied this problem. A recent, comprehensive literature review of the HVRP and its numerous variants, solution algorithms and applications can be found in [156, 170]. Among the 150 or so works reviewed in these papers, only a single work [250] has attempted to address the effect of uncertainty.

2.1.1 *Our Contributions*

In this chapter, we study the modeling and solution of the robust HVRP under customer demand uncertainty. The goal is to determine a single set of vehicle routes and associated fleet composition such that the total demand served on any route is less than the associated vehicle capacity, under any realization of the demands in a prespecified uncertainty set. Our work generalizes those of [137, 138] for the robust CVRP along multiple directions. First, our work addresses not only the CVRP, but also all variants of the HVRP and the Fleet Size and Mix vehicle routing problem that have been considered in the literature, as well as the Site Dependent and Multi-Depot vehicle routing problems. Second, we consider three new families of practically-relevant uncertainty sets in addition to the two considered in [137, 138] and discuss how each of these five sets can be constructed using historical data. While these sets are well known in the robust optimization community, they seem to be new in the context of vehicle routing.

Finally, we develop robust versions of classical local search moves. Specifically, we present data structures that allow efficient evaluation of the local moves and establish time and storage complexities of updating these structures. The proposed local search is both modular and efficient: on the one hand, it can be incorporated into any metaheuristic algorithm; on the other hand, its computational complexity is similar to that for the deterministic problem. We also present a robust integer programming model that provides lower bounds on the optimal solution and allows us to quantify the quality of the heuristic solutions. We elucidate, via an extensive computational study, the computa-

tional overhead of incorporating robustness, the quality of the lower bounds from the exact algorithm, as well as the tradeoff between robustness and costs for the considered uncertainty sets.

2.2 ROBUST HETEROGENEOUS VEHICLE ROUTING

An undirected graph $G = (V, E)$ with nodes $V = \{0, 1, \ldots, n\}$ and edges E is given. The node $0 \in V$ represents the depot, whereas each node $i \in V_C := V \setminus \{0\}$ represents a customer with demand $q_i \in \mathbb{R}_+$. The depot is equipped with a heterogeneous fleet of vehicles, which is composed of a set $K = \{1, \ldots, m\}$ of m different *vehicle types*. For each type $k \in K$, m_k vehicles are available, each of which has capacity Q_k. Furthermore, each vehicle of type $k \in K$ incurs a *fixed cost* $f_k \in \mathbb{R}_+$ if it is used and a *routing cost* $c_{ijk} \in \mathbb{R}_+$ if it traverses the edge $(i, j) \in E$.

A *route* is a simple cycle in G that passes through the depot. We represent a route by $R = (r_1, \ldots, r_{|R|})$, where $r_l \in V_C$ represents the l^{th} customer and $|R|$ the number of customers visited on route R. We use the notation $i \in R$ to indicate that customer i is visited on route R, that is, $i = r_l$ for some $l \in \{1, \ldots, |R|\}$. If route R is performed by a vehicle of type $k \in K$, then a cost equal to the sum of the routing costs and fixed cost associated with that vehicle type is incurred, namely $c(R, k) = f_k + \sum_{l=0}^{|R|} c_{r_l r_{l+1} k}$, where we have defined $r_0 = r_{|R|+1} = 0$.

When the customer demands are known precisely, a *set of routes* $\mathbf{R} = (R_1, \ldots, R_H)$ in conjunction with a *fleet composition vector* $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_H)$ is said to define a *feasible* HVRP solution $(\mathbf{R}, \boldsymbol{\kappa})$ if and only if the following conditions are satisfied:

- **(C1)** The routes $R_1, \ldots, R_H$ partition the customer set V_C. In other words, each customer is visited on exactly one route.
- **(C2)** The number of routes performed by vehicles of type k does not exceed their available number, that is, $\sum_{h=1}^{H} \mathbb{I}[\kappa_h = k] \leq m_k$ for all $k \in K$, where $\mathbb{I}[\mathcal{E}]$ is the indicator function that evaluates to 1 if the expression $\mathcal{E}$ is true and 0 otherwise.
- **(C3)** The capacities of all vehicles are respected, that is, $\sum_{i \in R_h} q_i \leq Q_{\kappa_h}$ for all $h \in \{1, \ldots, H\}$.

The cost of a feasible solution is defined to be the sum of the costs of its individual routes, $c(\mathbf{R}, \boldsymbol{\kappa}) = \sum_{h=1}^{H} c(R_h, \kappa_h)$. The goal of the HVRP is then to determine a feasible solution of minimum cost.

The HVRP model generalizes several VRP variants that have been studied in the literature [29]. The distinguishing characteristics these variants are summarized in Table 2.1.

1. The classical Capacitated VRP (CVRP), in which a homogeneous fleet of v vehicles of identical capacity Q are available at a central depot. The HVRP reduces to the CVRP if we set $m = 1$, $m_1 = v$, $Q_1 = Q$ and $f_1 = 0$ (*i.e.*, there are no fixed costs associated with the vehicles).

2. The HVRP with no fixed costs ($f_k = 0$ for all $k \in K$) but with vehicle-*dependent* routing costs, commonly denoted as the HVRPD.
3. The Fleet Size and Mix VRP with *fixed* costs, vehicle-*dependent* routing costs and in which an unlimited number of vehicles of each type is available ($m_k = n$ for all $k \in K$), commonly denoted as FSMFD.
4. The Fleet Size and Mix VRP with *fixed* costs, vehicle-independent routing costs ($c_{ijk_1} = c_{ijk_2}$ for all $(k_1, k_2) \in K \times K$ and $(i, j) \in E$) and in which an unlimited number of vehicles of each type is available ($m_k = n$ for all $k \in K$), commonly denoted as FSMF.
5. The Fleet Size and Mix VRP with no fixed costs ($f_k = 0$ for all $k \in K$), vehicle-*dependent* routing costs and in which an unlimited number of vehicles of each type is available ($m_k = n$ for all $k \in K$), commonly denoted as FSMD.
6. The Site Dependent VRP (SDVRP), in which each customer site $i \in V_C$ can only be visited by a subset of vehicle types $K_i \subseteq K$, representing site-specific constraints. There are no fixed costs ($f_k = 0$ for all $k \in K$) and the routing costs are vehicle-independent but site-dependent. In other words, the routing costs are defined as follows (where we have defined $K_0 = K$):

$$c_{ijk} = \begin{cases} \hat{c}_{ij} & \text{if } k \in K_i \cap K_j, \\ +\infty & \text{otherwise,} \end{cases} \quad \forall (i, j) \in E.$$

7. The Multi-Depot CVRP (MDVRP), in which a homogeneous fleet of vehicles are stationed at m distinct depots. The vehicle capacities are identical ($Q_k = Q$ for all $k \in K$), their number is unlimited ($m_k = n$ for all $k \in K$) and there are no fixed costs associated with their use ($f_k = 0$ for all $k \in K$). The routing costs are vehicle-independent and are represented by a $(n + m) \times (n + m)$ cost matrix $\hat{c}$, where $\hat{c}_{n+k\,j}$ represents the routing cost along the edge connecting the k^{th} depot and customer $j \in V_C$, for all $k \in K$. In other words, we have:

$$c_{ijk} = \begin{cases} \hat{c}_{n+k\,j} & \text{if } i = 0, \\ \hat{c}_{ij} & \text{otherwise,} \end{cases} \quad \forall (i, j) \in E,\ \forall k \in K.$$

In practice, it is often the case that the customer demands are not known precisely when the vehicles are to be dispatched. In such cases, one option is to replace the unknown demands by their 'nominal' values (e.g., by considering a historical sample average) and solve the original deterministic model. Unfortunately, this would often lead to situations in which the constructed vehicle routes 'fail' during their execution (*e.g.*, the vehicles might exceed their carrying capacity in a pickup problem or fail to deliver the demanded quantity in a delivery problem), particularly in situations that deviate from the nominal. To prevent the occurrence of such situations, we adopt a robust optimization approach

Table 2.1: Distinguishing characteristics of the problem variants studied in the literature.

Variant	Vehicle fleet	Fleet size	Fixed costs	Routing costs
CVRP	Homogeneous	Limited	Ignored	Independent
HVRPFD	Heterogeneous	Limited	Considered	Dependent
HVRPD	Heterogeneous	Limited	Ignored	Dependent
FSMFD	Heterogeneous	Unlimited	Considered	Dependent
FSMD	Heterogeneous	Unlimited	Considered	Independent
FSMF	Heterogeneous	Unlimited	Ignored	Dependent
SDVRP	Heterogeneous	Limited	Ignored	Dependent
MDVRP	Homogeneous	Unlimited	Ignored	Dependent

and assume that the customer demands can take *any* values from an *uncertainty set* $\mathcal{Q} \subseteq \mathbb{R}^n_+$. This uncertainty set should be chosen in a way that reflects the decision-maker's *a priori* confidence regarding the possible values that the customer demands may take. Whenever historical data is available, as is the case in practice, statistical models can be used to explicitly parameterize the size and shape of the uncertainty set over this *a priori* confidence level. We shall revisit this matter in Section 2.3. For now, we shall only assume, without loss of generality, that the uncertainty set is a non-empty, closed and bounded subset of $\mathbb{R}^n_+$.

For a given choice of the uncertainty set $\mathcal{Q}$, the goal in the *robust HVRP* is to determine a *robust feasible* solution of minimum cost. The solution $(\mathbf{R}, \boldsymbol{\kappa})$ is said to be robust feasible if and only if it satisfies conditions **(C1)**, **(C2)** and **(D3)**.

(D3) The capacities of all vehicles are respected under any realization of the customer demands from the uncertainty set, that is, $\sum_{i \in R_h} q_i \leq Q_{\kappa_h}$ for all $h \in \{1, \ldots, H\}$ and all $q \in \mathcal{Q}$.

By construction, the condition **(D3)** is equivalent to verifying if $\max_{q \in \mathcal{Q}} \sum_{i \in R_h} q_i \leq Q_{\kappa_h}$ is satisfied for each route $h \in \{1, \ldots, H\}$. In other words, the total load carried by any vehicle must be less than its capacity under the worst-case realization of the customer demands from the uncertainty set $\mathcal{Q}$. Efficiently evaluating the worst-case load, and hence, verifying condition **(D3)**, is key to developing efficient (exact and heuristic) algorithms for the robust HVRP. The efficient computation of the worst-case load is the subject of the next section, whereas its integration into heuristic and exact methods is discussed in Sections 2.4 and 2.5, respectively.

2.3 EFFICIENT COMPUTATION OF THE WORST-CASE LOAD

Given a vehicle route R, we define its *worst-case load* as the optimal value of the following problem:

$$\max_{q \in \mathcal{Q}} \sum_{i \in R} q_i. \tag{2.1}$$

First, note that we can assume, without loss of generality, that the uncertainty set $\mathcal{Q}$ is convex. This is because the objective function of problem (2.1) is linear and therefore, we can equivalently replace the feasible region $\mathcal{Q}$ with its convex hull. Therefore, in general, computing the worst-case load of a route requires the solution of a convex optimization problem. Even if $\mathcal{Q}$ is a polyhedron, say in N dimensions and described by M inequalities, then computing an optimal solution of problem (2.1) requires $\mathcal{O}(MN^2)$ arithmetic operations using interior point methods [62].

As we shall see later in Sections 2.4 and 2.5, this is extremely slow when used in the context of a heuristic or an exact method where hundreds or even thousands of routes are constructed *every second*, and their associated worst-case loads must be computed in order to verify if condition **(D3)** is satisfied. It would be impractical to call a general-purpose convex optimization solver to solve problem (2.1) in such cases. Furthermore, the successive routes constructed by a heuristic method differ only marginally; therefore, if we know the worst-case load of a route R, then we would like to compute the worst-case load of routes R' which are "almost similar" to R with minimal additional effort. It is not clear how this can be achieved using a general-purpose solver.

Fortunately, it can be shown that the worst-case load can be efficiently computed for a broad class of popular but practically-relevant uncertainty sets. Section 2.3.1 elaborates on the structure of these sets, while Section 2.3.2 illustrates how the worst-case load can be efficiently in such cases.

2.3.1 *Uncertainty Sets*

We note that if the uncertainty set is rectangular; that is, $\mathcal{Q} = \{q_1 : q \in \mathcal{Q}\} \times \ldots \times \{q_n : q \in \mathcal{Q}\}$, then the optimal solution of problem (2.1) is attained when each customer demand attains its worst realization individually (defined as $\bar{q}_i = \max\{q_i : q \in \mathcal{Q}\}$), irrespective of the other customer demands. However, this is a very conservative choice as it is unlikely that all customer demands will simultaneously attain their maximum possible values. Therefore, we only consider those cases where the uncertainty set is not rectangular.

2.3.1.1 *Budget sets*

Consider the uncertainty set of the following form (see also Figure 2.1):

$$\mathcal{Q}_B = \left\{ q \in [\underline{q}, \bar{q}] : \sum_{i \in B_l} q_i \leq b_l \quad \forall l \in \{1, \ldots, L\}. \right\} \tag{2.2}$$

This uncertainty set is formed by intersecting the n-dimensional hyperrectangle $[\underline{q}, \bar{q}]$ with the $L \in \mathbb{N}$ budget constraints involving customer subsets $B_l \subseteq V_C$. The l^{th} budget constraint imposes a limit $b_l \in \mathbb{R}_+$ on the cumulative demand of the customers in the set B_l. Observe that, by setting $b_l = \sum_{i \in B_l} \underline{q}_i$, for all $l \in \{1, \ldots, L\}$, the uncertainty set reduces to a singleton $\mathcal{Q}_B = \{\underline{q}\}$, whereas by setting $b_l = \sum_{i \in B_l} \bar{q}_i$, the uncertainty set becomes the n-dimensional hyperrectangle $[\underline{q}, \bar{q}]$. To exclude empty sets, we shall assume, without loss of generality, that $\underline{q} \leq \bar{q}$ and $\sum_{i \in B_l} \underline{q}_i \leq b_l$.

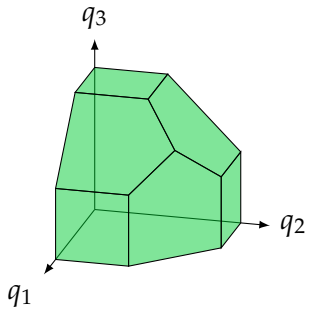


Figure 2.1: Example of a budget uncertainty set with $L = 3$ budget constraints.

The budget set $\mathcal{Q}_B$ reflects the belief that the customer demand q_i can individually vary between $\underline{q}_i$ and $\bar{q}_i$, but the cumulative customer demand over various subsets cannot exceed a certain limit. Intuitively, this belief is rooted in the fact that unless the customer demands exhibit perfect correlations, it is unlikely that they will attain their maximum values simultaneously. Statistically, budget sets are motivated from limit laws of probability, such as the *central limit theorem*. Indeed, if the customer demands are independent random variables with means q_i^0 and variances σ_i^2, for $i \in V_C$, then under mild technical conditions, the Lyapunov central limit theorem implies that for sufficiently large $|B_l|$, the sum of the normalized customer demands in the set B_l converges in distribution to a standard normal random variable. Stated differently, the inequality

$$\sum_{i \in B_l} q_i \leq \sum_{i \in B_l} q_i^0 + \Phi^{-1}(\gamma) s_l, \quad \text{where } s_l^2 = \sum_{i \in B_l} \sigma_i^2,$$

is satisfied with probability $\gamma \in (0, 1)$, where $\Phi^{-1}(\cdot)$ denotes the inverse cumulative distribution function of the standard normal random variable. One can verify that this inequality can be incorporated as a budget constraint in the uncertainty set $\mathcal{Q}_B$. Thus, we can use standard statistical tools to estimate q_i^0 and σ, and control the size of the uncertainty set using the probability level γ. The shape of the uncertainty set can

be controlled by selecting appropriate customer subsets B_l. For example, these could represent geographical regions such as municipalities, counties or states.

In the most general case, computing the worst-case load (2.1) for the budget uncertainty set amounts to solving a *fractional packing problem*, for which a $(1+\epsilon)$-approximate solution can be computed in $\mathcal{O}(\epsilon^{-2}Ln)$ time, assuming $L \geq n$ [274]. A much more efficient computation of the worst-case load is possible if we make the additional assumption that the customer subsets B_l are pairwise *disjoint*, that is, $B_l \cap B_{l'} = \emptyset$ for all $l \neq l'$. In the remainder of the chapter, we shall therefore assume that this additional requirement is satisfied.

2.3.1.2 *Factor models*

Consider the uncertainty set of the following form (see also Figure 2.2):

$$\mathcal{Q}_F = \left\{q \in \mathbb{R}^n : q = q^0 + \Psi\xi \text{ for some } \xi \in \Xi_F\right\}, \text{where } \Xi_F = \left\{\xi \in [-1,1]^F : \left|e^\top \xi\right| \leq \beta F\right\}. \tag{2.3}$$

Here, $q^0 \in \mathbb{R}^n_+$, $F \in \mathbb{N}$, $\Psi \in \mathbb{R}^{n\times F}$ and $\beta \in [0,1]$ are parameters that need to be specified by the modeler. Note that $e \in \mathbb{R}^F$ denotes the vector of ones.

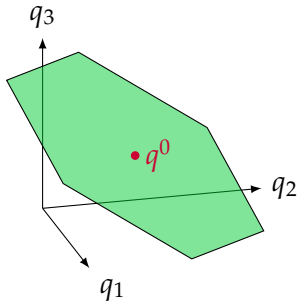


Figure 2.2: Example of a factor model uncertainty set with $F = 2$ factors.

The uncertainty set (2.3) stipulates that the unknown customer demands q are distributed around a nominal demand vector q^0, subject to an additive disturbance of $\Psi\xi$. This disturbance is a linear combination of independent factors $\xi_1, \ldots, \xi_F$ that reside in the F-dimensional hypercube. Typically, one has $F \ll n$, so that the linear operator Ψ allows us to model correlations in the (possibly high dimensional space of) customer demands through correlations in the low-dimensional space of the factors. The matrix Ψ is also known as the *factor loading matrix* and whenever historical data is available, it can be constructed using statistical tools such as principal components analysis or factor analysis. The constraint $\left|e^\top \xi\right| \leq \beta F$ reflects the belief that not all of the independent factors can simultaneously attain their extreme values. For example, setting $\beta = 0$ will enforce that as many factors will be above 0 as there will be below 0, and the resulting factor model has also been referred to as "zero-net-alpha adjustment" in portfolio

optimization [76]. Similarly, setting $\beta = 1$ will reduce Ξ_F to an F-dimensional hypercube. In general, whenever historical data is available, an appropriate value of β can be chosen by combining a central limit law or tail bound with an *a priori* confidence level, as discussed in Section 2.3.1.1.

2.3.1.3 *Ellipsoidal sets*

Consider the uncertainty set of the following form (see also Figure 2.3):

$$\mathcal{Q}_E = \left\{ q \in \mathbb{R}^n : q = q^0 + \Sigma^{1/2}\xi \text{ for some } \xi \in \Xi_E \right\}, \text{where } \Xi_E = \left\{ \xi \in \mathbb{R}^n : \xi^\top \xi \leq 1 \right\}. \tag{2.4}$$

Here, $q^0 \in \mathbb{R}^n_+$ and $\Sigma \in \mathbb{S}^n_+$ are parameters that need to be specified by the modeler, and $\mathbb{S}^n_+$ denotes the cone of symmetric positive semidefinite matrices. Whenever Σ is a diagonal matrix, we shall refer to the resulting uncertainty set as an *axis-parallel* ellipsoidal set.

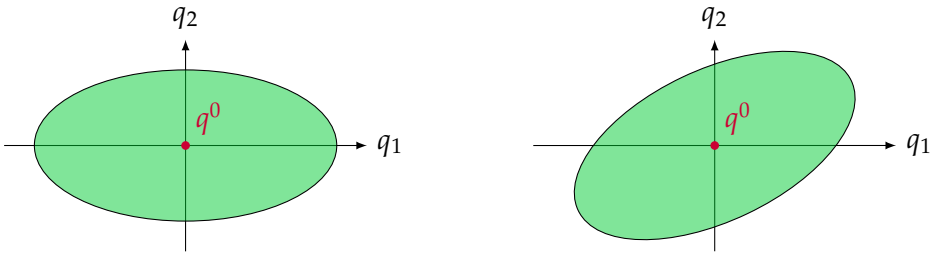


Figure 2.3: Example of an axis-parallel (left) and general (right) ellipsoidal set.

The uncertainty set (2.4) stipulates that the customer demands can only attain values in an ellipsoid centered around a nominal demand vector q^0. We note that if the matrix Σ is singular, then this ellipsoid is degenerate. Otherwise, the set $\mathcal{Q}_E$ can be equivalently represented as follows:

$$\mathcal{Q}_E = \left\{ q \in \mathbb{R}^n : \left(q - q^0\right)^\top \Sigma^{-1} \left(q - q^0\right) \leq 1 \right\}.$$

This uncertainty set reflects the belief that the customer demand vector q is a multivariate normal random variable with (unknown) mean μ and (unknown) covariance matrix Σ^{true}. The resulting ellipsoid can therefore be identified as a confidence region for the unknown mean μ. Stated differently, suppose $D > n$ records of historical data are available, and q^0 and Σ denote the associated sample mean and (unbiased) sample covariance, respectively. Then, the inequality,

$$\left(\mu - q^0\right)^\top \Sigma^{-1} (\mu - q^0) \leq \frac{n}{D-n} \frac{D-1}{D} H^{-1}_{n;D-n}(\gamma),$$

is satisfied with probability $\gamma \in (0,1)$, where $H^{-1}_{n;D-n}$ denotes the inverse cumulative distribution function of the F-distribution with parameters n and $D - n$. Thus, we can control the size of the ellipsoidal uncertainty set using the probability level γ.

2.3.1.4 *Cardinality-constrained sets*

Consider the uncertainty set of the form (see also Figure 2.4):

$$\mathcal{Q}_G = \left\{q \in [q^0, q^0 + \hat{q}] : q = q^0 + (\hat{q} \circ \xi) \text{ for some } \xi \in \Xi_G\right\}, \\ \text{where } \Xi_G = \left\{\xi \in [0,1]^n : e^\top \xi \leq \Gamma\right\}. \tag{2.5}$$

Here, $q^0 \in \mathbb{R}^n_+$, $\hat{q} \in \mathbb{R}^n_+$ and $\Gamma \in [0, n]$ are parameters that need to be specified by the modeler. Note that $e \in \mathbb{R}^n$ denotes the vector of ones and $(\hat{q} \circ \xi) \in \mathbb{R}^n$ denotes the Hadamard product between vectors $\hat{q}$ and ξ; that is, $(\hat{q} \circ \xi)_i = \hat{q}_i \xi_i$ for all $i = 1, \ldots, n$.

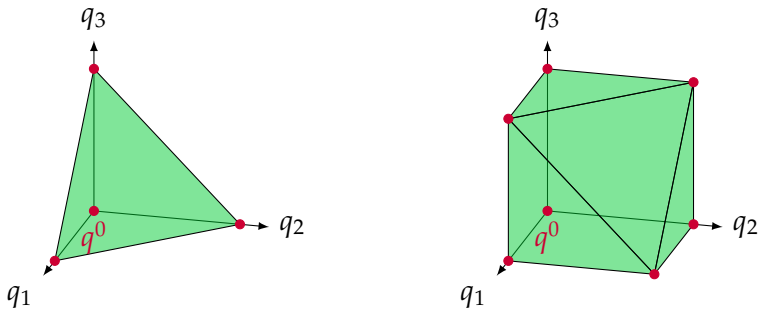


Figure 2.4: Example of a cardinality-constrained set with $\Gamma = 1$ (left) and $\Gamma = 2$ (right).

The uncertainty set stipulates that each customer demand q_i can deviate from its nominal value q^0_i by up to $\hat{q}_i$. However, the total number of demands that can simultaneously deviate from their nominal values is bounded by $\lceil \Gamma \rceil$; of these, $\lfloor \Gamma \rfloor$ customer demands can maximally deviate up to $\hat{q}$ while one customer demand can deviate up to $(\Gamma - \lfloor \Gamma \rfloor)\hat{q}_i$. For example, if we set $\Gamma = 0$, then the uncertainty set reduces to a singleton $\mathcal{Q}_G = \{q^0\}$, whereas if we set $\Gamma = n$, then the uncertainty set becomes the n-dimensional hyperrectangle $[q^0, q^0 + \hat{q}]$. Observe that Ξ_G is the convex hull of the set $\{\xi \in [0,1]^n : \|\xi\|_0 \leq \Gamma\}$, where $\|\xi\|_0$ counts the number of non-zero elements in ξ. Therefore, the inequality $e^\top \xi \leq \Gamma$ may be interpreted as constraining the number of elements which may simultaneously deviate from their nominal values, which explains the name "cardinality-constrained". This uncertainty set was originally proposed in [57] and is also popularly referred to as a "budgeted" or "gamma" uncertainty set.

Using a similar argument as in [57], it can be shown that if the demand of each customer $i \in V_C$ is a symmetric and bounded random variable $\tilde{q}_i$ that takes values in $\left[q^0_i - \hat{q}_i, q^0_i + \hat{q}_i\right]$, then the *actual* worst-case load of any route R satisfies

$$\text{Probability}\left(\sum_{i \in R} \tilde{q}_i \leq \max_{q \in \mathcal{Q}_G} \sum_{i \in R} q_i\right) \geq 1 - \exp(-\Gamma^2/2n).$$

In other words, if up to Γ customer demands actually deviate from their nominal values, then the actual worst-case load of route R is bounded by the quantity $\max_{q \in \mathcal{Q}_G} \sum_{i \in R} q_i$,

whereas even if more than Γ customer demands deviate from their nominal values, then the actual worst-case load is bounded with very high probability. We note that a tighter probabilistic bound has been established in [57]. In practice, one can use any of these bounds to determine a suitable value of Γ.

2.3.1.5 *Discrete sets*

Consider the uncertainty set of the following form (see also Figure 2.5):

$$\mathcal{Q}_D = \text{conv}\left\{q^{(j)} : j = 1, \ldots, D\right\}, \tag{2.6}$$

where $\text{conv}(\cdot)$ denotes the convex hull of a finite set of points. Here, $q^{(1)}, \ldots q^{(D)} \in \mathbb{R}^n_+$ are $D \in \mathbb{N}$ distinct realizations of the uncertain customer demands that need to be specified by the modeler.

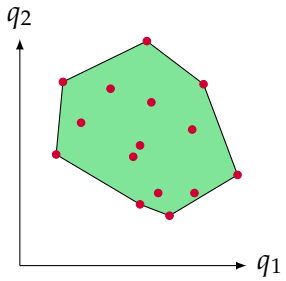


Figure 2.5: Example of a discrete uncertainty set with $D = 15$ data points.

The uncertainty set stipulates that the customer demands will take values inside the convex hull of D *a priori* specified demand vectors in n-dimensional space. This set is expected to be meaningful only if sufficient historical records are available; that is, if D is sufficiently large. However, care must be taken to discard statistical outliers in order to avoid becoming overly risk-averse. More generally, one may want to consider only a small subset of all historically observed realizations by uniformly sampling a fraction $\gamma \in (0, 1)$ of the available realizations.

2.3.2 *Closed-Form Expressions of the Worst-Case Load*

The key result of this section is that the worst-case load of a vehicle route can be computed in closed-form for each of the five classes of uncertainty sets described in the previous section. This formalized in Proposition 2.1.

Proposition 2.1. *Suppose R is a given route and $S \subseteq V_C$ is the set of customers visited on R. Then, the worst-case load of route R over the uncertainty sets $\mathcal{Q}_B$, $\mathcal{Q}_F$, $\mathcal{Q}_E$, $\mathcal{Q}_G$ and $\mathcal{Q}_D$ is as follows.*

Table 2.2: Closed-form expressions of the worst-case load of a vehicle route.

$\mathcal{Q}$	$\max_{q \in \mathcal{Q}} \sum_{i \in R} q_i$
$\mathcal{Q}_B$	$\sum_{i \in S} \bar{q}_i - \sum_{l=1}^{L} \max \left\{ 0, \sum_{i \in S \cap B_l} (\bar{q}_i - \underline{q}_i) - \left(b_l - \sum_{i \in B_l} \underline{q}_i \right) \right\}$ (2.7)
$\mathcal{Q}_F$	$\sum_{i \in S} q_i^0 - \min \left\{ \sum_{f=1}^{F} \sum_{i \in S} \Psi_{if} - \lambda + \beta F \lvert\lambda\rvert : \lambda \in \left\{ 0, \sum_{i \in S} \Psi_{if_{\ell^+}}, \sum_{i \in S} \Psi_{if_{\ell^-}} \right\} \right\}$, (2.8)
	where $f_1, \dots f_F$ *represents an ordering of the factors such that* $\sum_{i \in S} \Psi_{if_1} \geq \dots \geq \sum_{i \in S} \Psi_{if_F}$, *and* $\ell^+ = \lceil (1+\beta)F/2 \rceil$, $\ell^- = \max\{1, \lceil (1-\beta)F/2 \rceil\}$.
$\mathcal{Q}_E$	$\sum_{i \in S} q_i^0 + \left\| \sum_{i \in S} \Sigma_i^{1/2} \right\|_2$, (2.9)
	where $\Sigma_i^{1/2}$ *denotes the* i^{th} *column of* $\Sigma^{1/2}$ *and* $\|\cdot\|_2$ *denotes the* ℓ_2*-norm of a vector.*
$\mathcal{Q}_G$	$\sum_{i \in S} q_i^0 + \sum_{\ell=1}^{\min\{\lvert S\rvert, \lfloor \Gamma \rfloor\}} \hat{q}_{g_\ell} + \lambda$, (2.10)
	where $g_1, \dots g_{\lvert S\rvert}$ *represents an ordering of the customers in* S *such that* $\hat{q}_{g_1} \geq \dots \geq \hat{q}_{g_{\lvert S\rvert}}$, *and* $\lambda = (\Gamma - \lfloor \Gamma \rfloor)\, \hat{q}_{g_{\lfloor \Gamma \rfloor + 1}}$, *if* $\lvert S\rvert \geq \lfloor \Gamma \rfloor + 1$ *and* 0 *otherwise.*
$\mathcal{Q}_D$	$\max \left\{ \sum_{i \in S} q_i^{(d)} : d = 1, \dots, D \right\}$ (2.11)

Proof. The validity of the expressions for $\mathcal{Q} = \mathcal{Q}_B$ and $\mathcal{Q} = \mathcal{Q}_F$ has been shown in [137].

Suppose that $\mathcal{Q} = \mathcal{Q}_E$. In this case, the worst-case problem (2.1) can be reformulated as follows:

$$\sum_{i \in S} q_i^0 + \max_{\xi \in \mathbb{R}^n} \left\{ \sum_{i \in S} \xi^\top \Sigma_i^{1/2} : \xi^\top \xi \leq 1 \right\}.$$

The above maximization problem is a convex optimization problem that satisfies Slater's constraint qualification (e.g., $0 \in \mathbb{R}^n$ is strictly feasible). Therefore, the Karush-Kuhn-Tucker conditions are both necessary and sufficient to characterize its optimal solution. This leads to expression (2.9).

Suppose that $\mathcal{Q} = \mathcal{Q}_G$. In this case, the worst-case problem (2.1) can be reformulated as follows:

$$\sum_{i \in S} q_i^0 + \max_{\xi \in \mathbb{R}^n_+} \left\{ \sum_{i \in S} \hat{q}_i \xi_i : \xi_i \leq 1 \ \forall i = 1, \dots, n, \quad \sum_{i=1}^{n} \xi_i \leq \Gamma \right\}.$$

The above maximization problem is an instance of a *fractional knapsack* problem with n items, each with unit weight; the value of item i is $\hat{q}_i \mathbb{I}[i \in S]$. It is well known (e.g.,

see [97]) that this problem can be solved by a greedy algorithm that considers the items in non-increasing order of their values per unit weight, *i.e.*, in non-increasing order of the $\hat{q}$ values of the items in S. A straightforward application of this greedy algorithm leads to expression (2.10).

Suppose now that $\mathcal{Q} = \mathcal{Q}_D$. Since $\mathcal{Q}_D$ is a bounded polytope and the objective function of the worst-case problem (2.1) is linear, the optimal solution is attained at a vertex of $\mathcal{Q}_D$. This leads to expression (2.11) since the set of vertices of $\mathcal{Q}_D$ is a subset of the D points that parametrize it. □

Proposition 2.1 suggests that the worst-case load of a route can be computed much faster by evaluating the associated closed-form expressions than by invoking a general-purpose optimization solver. Furthermore, if we know the worst-case load of a route R that visits a certain subset of customers $S \subseteq V_C$, and we would like to calculate the worst-case load of a route R' that visits exactly one additional or fewer customer $i \in V_C$, that is, if R' visits $S \cup \{i\}$ or $S \setminus \{i\}$, then the worst-case load can be *incrementally updated* even faster by using appropriate data structures. This is formalized in Proposition 2.2.

Proposition 2.2. *The following time and storage complexities can be achieved for computing the worst-case load of route that visits a set of customers S. Here, "incremental update" refers to the complexity of updating the worst-case load when a single customer is added to or removed from S.*

Table 2.3: Time and storage complexities for computing the worst-case load of a vehicle route.

$\mathcal{Q}$	From scratch	Incremental update	
	Time	Time	Storage
$\mathcal{Q}_B$	$\mathcal{O}(\lvert S\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(L)$
$\mathcal{Q}_F$	$\mathcal{O}(\lvert S\rvert F + F \log F)$	$\mathcal{O}(F \log F)$	$\mathcal{O}(F)$
$\mathcal{Q}_E$ (axis-parallel)	$\mathcal{O}(\lvert S\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\mathcal{Q}_E$ (general)	$\mathcal{O}(\lvert S\rvert n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
$\mathcal{Q}_G$	$\mathcal{O}(\lvert S\rvert)$	$\mathcal{O}(\log(\lvert S\rvert))^{\text{a}}$	$\mathcal{O}(n)^{\text{b}}$
$\mathcal{Q}_D$	$\mathcal{O}(\lvert S\rvert D)$	$\mathcal{O}(D)$	$\mathcal{O}(D)$

[a,b] These can be optimized to $\mathcal{O}(\log(\Gamma))$ and $\mathcal{O}(\Gamma)$ respectively, if customers are only added to S.

Proof. • Consider $\mathcal{Q} = \mathcal{Q}_B$. The time complexity for the "from scratch" computation follows directly from expression (2.7). To perform incremental updates, we store the following quantities along with the customer set S: $\pi = \sum_{i \in S} \bar{q}_i$, $\rho_l = \sum_{i \in S \cap B_l} \left(\bar{q}_i - \underline{q}_i\right) - \left(b_l - \sum_{i \in B_l} \underline{q}_i\right)$ for all $l = 1, \ldots, L$, and $z = \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i$. Initially, when $S = \emptyset$, we have $(\pi, \rho_l, z) = (0, -(b_l - \sum_{i \in B_l} \underline{q}_i), 0)$. Now, suppose that we would like to calculate the worst-case load of a route visiting the customer set $S' = S \cup \{j\}$, where customer j participates in

the budget l_j; that is, $j \in B_{l_j}$. Then, we would perform the following $\mathcal{O}(1)$ update: $\pi^{\text{new}} \leftarrow \pi^{\text{old}} + \bar{q}_j$, $\rho_l^{\text{new}} \leftarrow \rho_l^{\text{old}} + (\bar{q}_j - \underline{q}_j)$, $z^{\text{new}} \leftarrow z^{\text{old}} + (\pi^{\text{new}} - \pi^{\text{old}}) - ([\rho_{l_j}^{\text{new}}]_+ - [\rho_{l_j}^{\text{old}}]_+)$, where $[\cdot]_+ = \max\{\cdot, 0\}$. If customer j does not participate in any budget, then we would repeat the same steps except ρ is not updated. A similar update applies when $S' = S \setminus \{j\}$ for some $j \in S$. We do not present this for the sake of brevity.

• Consider $\mathcal{Q} = \mathcal{Q}_F$. The time complexity for the "from scratch" computation follows from the $\mathcal{O}(|S|)$ time to calculate $\sum_{i \in S} \Psi_{if}$ for each of the F factors and the $\mathcal{O}(F \log F)$ time to compute the ordering $f_1, \ldots, f_F$. To perform incremental updates, we store the following quantities along with the customer set S: $\pi = \sum_{i \in S} q_i^0$, $\rho_f = \sum_{i \in S} \Psi_{if}$ for all $f = 1, \ldots, F$, and $z = \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i$. Initially, when $S = \emptyset$, we have $(\pi, \rho_f, z) = (0, 0, 0)$. Now, suppose that we would like to calculate the worst-case load of a route visiting the customer set $S' = S \cup \{j\}$. Then, we would perform the following $\mathcal{O}(F \log F)$ update: $\pi^{\text{new}} \leftarrow \pi^{\text{old}} + q_j^0$, $\rho_f^{\text{new}} \leftarrow \rho_f^{\text{old}} + \Psi_{jf}$ for each $f = 1, \ldots, F$, and $z^{\text{new}} \leftarrow \pi^{\text{new}} + \sum_{f=1}^{F} \xi_f^{\text{wc}} \rho_{f_F}^{\text{new}}$, where $f_1, \ldots, f_F$ is an ordering of the factors according to $\rho_{f_1} \geq \ldots \geq \rho_{f_F}$ and $\xi^{\text{wc}} \in \Xi_F$ is defined by computing $M = \sum_{f=1}^{F} \mathbb{I}[\rho_f^{\text{new}} \geq 0]$ and $N = F - M$, and then:

Check	F'	τ	ς	ξ^{wc}	
$M \geq N$	$M - N$	$\lfloor (F' - \lfloor \beta F' \rfloor)/2 \rfloor$	$\lfloor \beta F' \rfloor + \tau$	$(e^N, e^\varsigma, -e^\tau, -e^N)$	if $\varsigma + \tau = F'$
				$(e^N, e^\varsigma, \beta F - \lfloor \beta F \rfloor, -e^\tau, -e^N)$	if $\varsigma + \tau \neq F'$
$M < N$	$N - M$			$(e^M, e^\tau, -e^\varsigma, -e^M)$	if $\varsigma + \tau = F'$
				$(e^M, e^\tau, -\beta F + \lfloor \beta F \rfloor, -e^\varsigma, -e^M)$	if $\varsigma + \tau \neq F'$

Here, e^{dim} denotes the vector of ones in $\mathbb{R}^{\text{dim}}$. A similar update applies when $S' = S \setminus \{j\}$ for some $j \in S$.

• Consider $\mathcal{Q} = \mathcal{Q}_E$, where $\mathcal{Q}_E$ is axis-parallel; that is, $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ is a diagonal matrix. In this case, the expression (2.9) simplifies to $\sum_{i \in S} q_i^0 + \sqrt{\sum_{i \in S} \sigma_i^2}$, which can be computed in $\mathcal{O}(|S|)$ time. To perform incremental updates, we store the following quantities along with the customer set S: $\pi = \sum_{i \in S} q_i^0$, $\rho = \sum_{i \in S} \sigma_i^2$ and $z = \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i$. Initially, when $S = \emptyset$, we have $(\pi, \rho, z) = (0, 0, 0)$. Now, to calculate the worst-case load of a route visiting the customer set $S' = S \cup \{j\}$, we would perform the following $\mathcal{O}(1)$ update: $\pi^{\text{new}} \leftarrow \pi^{\text{old}} + q_j^0$, $\rho^{\text{new}} \leftarrow \rho^{\text{old}} + \sigma_j^2$, $z^{\text{new}} \leftarrow \pi^{\text{new}} + \sqrt{\rho^{\text{new}}}$. A similar update applies when $S' = S \setminus \{j\}$ for some $j \in S$.

• Consider now $\mathcal{Q} = \mathcal{Q}_E$, where Σ is a general matrix. In this case, expression (2.9) can be written as $\sum_{i \in S} q_i^0 + \sqrt{\sum_{j=1}^{n} \left(\sum_{i \in S} \Sigma_{ij}^{1/2} \right)^2}$, which can be computed in $\mathcal{O}(|S|\, n)$ time. To perform incremental updates, we store the following quantities along with the customer set S: $\pi = \sum_{i \in S} q_i^0$, $\rho_l = \sum_{i \in S} \Sigma_{il}^{1/2}$ for all $l = 1, \ldots, n$, and $z = \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i$. When $S = \emptyset$, we have $(\pi, \rho_l, z) = (0, 0, 0)$. To calculate the worst-case load of a route visiting the customer set $S' = S \cup \{j\}$, we would perform the following $\mathcal{O}(n)$ update: $\pi^{\text{new}} \leftarrow \pi^{\text{old}} + q_j^0$, $\rho_l^{\text{new}} \leftarrow \rho_l^{\text{old}} + \Sigma_{jl}^{1/2}$ for all $l = 1, \ldots, n$, $z^{\text{new}} \leftarrow \pi^{\text{new}} + \sqrt{\sum_{l=1}^{n} \left(\rho_l^{\text{new}}\right)^2}$. A similar update applies when $S' = S \setminus \{j\}$ for some $j \in S$.

• Consider $\mathcal{Q} = \mathcal{Q}_G$. An examination of expression (2.10) reveals that we do not need to sort the customers in S with respect to their $\hat{q}$ values. Instead, we only need to identify the subset of customers with the $(\lfloor\Gamma\rfloor + 1)^{\text{th}}$ largest $\hat{q}$ values. This can be achieved in $\mathcal{O}(|S|)$ time with a *partition-based selection algorithm* such as quickselect with an appropriate pivoting strategy (e.g., see [90]). To perform incremental updates, we store the following quantities along with the customer set S: $\pi = \sum_{i\in S} q_i^0$, $s = \min\{|S|, \lfloor\Gamma\rfloor\}$, array $h^+ = [\hat{q}_{g_1}, \ldots, \hat{q}_{g_s}]$ implemented as a binary min-heap, array $h_- = [\hat{q}_{g_{s+2}}, \ldots, \hat{q}_{g_{|S|}}]$ implemented as a binary max-heap, $\rho_+ =$ sum of entries of h_+, $\rho_0 = \hat{q}_{g_{s+1}}$, and $z = \max_{q\in\mathcal{Q}} \sum_{i\in S} q_i$. Define $h_- = \emptyset$ if $|S| \leq \lfloor\Gamma\rfloor + 1$ and $\rho_0 = 0$ if $s + 1 > |S|$. When $S = \emptyset$, we have $(\pi, s, h_+, h_-, \rho_+, \rho_0, z) = (0, 0, \emptyset, \emptyset, 0, 0, 0)$. To calculate the worst-case load of a route visiting the customer set $S' = S \cup \{j\}$, we perform the following updates: if $s^{\text{old}} < \lfloor\Gamma\rfloor$, then $s^{\text{new}} \leftarrow s^{\text{old}} + 1$, $h_+^{\text{new}} \leftarrow h_+^{\text{old}}.\text{insert}(\hat{q}_j)$, $\rho_+^{\text{new}} \leftarrow \rho_+^{\text{old}} + \hat{q}_j$; else if $\hat{q}_j \leq \rho_0$, then $h_-^{\text{new}} \leftarrow h_-^{\text{old}}.\text{insert}(\hat{q}_j)$; else if $\hat{q}_j < \min(h_+^{\text{old}})$, then $\rho_0^{\text{new}} \leftarrow \hat{q}_j$, $h_-^{\text{new}} \leftarrow h_-^{\text{old}}.\text{insert}(\rho_0^{\text{old}})$; else, $h_+^{\text{new}} \leftarrow h_+^{\text{old}}.\text{delete}(\min(h_+^{\text{old}})).\text{insert}(\hat{q}_j)$, $\rho_+^{\text{new}} \leftarrow \rho_+^{\text{old}} + \hat{q}_j - \min(h_+^{\text{old}})$, $\rho_0^{\text{new}} \leftarrow \min(h_+^{\text{old}})$ and $h_-^{\text{new}} \leftarrow h_-^{\text{old}}.\text{insert}(\rho_0^{\text{old}})$. In all cases, we also update $\pi^{\text{new}} \leftarrow \pi^{\text{old}} + q_j^0$ and $z^{\text{new}} \leftarrow \pi^{\text{new}} + \rho_+^{\text{new}} + (\Gamma - \lfloor\Gamma\rfloor)\rho_0^{\text{new}}$. The overall time complexity is $\mathcal{O}(|S|)$ and is dictated by the insertion and deletion operations in the heaps h_+ and h_-. Note that h_- was not used in determining the value of z. In fact, it is used only when updating the worst-case load after a deletion has occurred. If the quantities are maintained only by the addition of elements into initially empty structures, then h_+ is sufficient and since its size is at most $\lfloor\Gamma\rfloor$, the overall time and storage complexities would be $\mathcal{O}(\log(\Gamma))$ and $\mathcal{O}(\Gamma)$ respectively.

• Consider now $\mathcal{Q} = \mathcal{Q}_D$. The time complexity for the "from scratch" computation follows directly from expression (2.11). To perform incremental updates, we store the following quantities along with the customer set S: $\rho_d = \sum_{i\in S} q_i^{(d)}$ for all $d = 1, \ldots, D$, and $z = \max_{q\in\mathcal{Q}} \sum_{i\in S} q_i$. When $S = \emptyset$, we have $(\rho_d, z) = (0, 0)$. To calculate the worst-case load of a route visiting the customer set $S' = S \cup \{j\}$, we would perform the following $\mathcal{O}(D)$ updates: $\rho_d^{\text{new}} \leftarrow \rho_d^{\text{old}} + q_j^{(d)}$ for all $d = 1, \ldots, D$, $z^{\text{new}} \leftarrow \max\{\rho_d^{\text{new}} : d = 1, \ldots, D\}$. A similar update applies when $S' = S \setminus \{j\}$ for some $j \in S$. □

2.4 ROBUST LOCAL SEARCH AND METAHEURISTICS

The vast majority of metaheuristics for solving large-scale instances of (deterministic) vehicle routing problems are all based on *local search*. Section 2.4.1 illustrates how the results from the previous section allow us to efficiently extend local search to the robust setting. This robust version of local search can then be incorporated in a modular fashion into any metaheuristic algorithm. To illustrate this, Section 2.4.2 provides details on an Iterated Local Search (ILS) metaheuristic, while Section 2.4.3 provides details on an Adaptive Memory Programming (AMP) metaheuristic.

2.4.1 *Robust Local Search*

The basis of all local search methods is the repeated use of a set of elementary moves that transform a *current solution* $(\mathbf{R}, \boldsymbol{\kappa})$ into a different, *neighbor solution*. The set of all solutions that can be reached from the current one using a set Y of moves is called the neighborhood of the current solution with respect to the move set, $\Omega_Y(\mathbf{R}, \boldsymbol{\kappa})$. The two major building blocks of all local search methods are therefore, the definition of the *neighborhoods*, and the exploration of the neighborhoods using a *search algorithm*. The following paragraphs describe some classical *node-exchange* and *edge-exchange* neighborhoods that involve the deletion and re-insertion of nodes or edges [1, 120]. We note however, that our results readily generalize to other neighborhoods as well (*e.g.*, λ-interchange, ejection chains etc.).

RELOCATE NEIGHBORHOOD. The classical *relocate* neighborhood (also known as *1-0 relocate*) consists of the set $\Omega_{\text{rel}}(\mathbf{R}, \boldsymbol{\kappa})$ of all solutions that can be obtained by relocating a single customer from its current position in $\mathbf{R}$ to a new position. If the new position lies on the same route as the current one, then the corresponding move is known as an intra-route relocate move, whereas if the new position lies on a different route, then it is referred to as an inter-route relocate or *shift(1,0)* move, see Figure 2.6. The size of this neighborhood $|\Omega_{\text{rel}}(\mathbf{R}, \boldsymbol{\kappa})|$ is $\mathcal{O}(n^2)$.

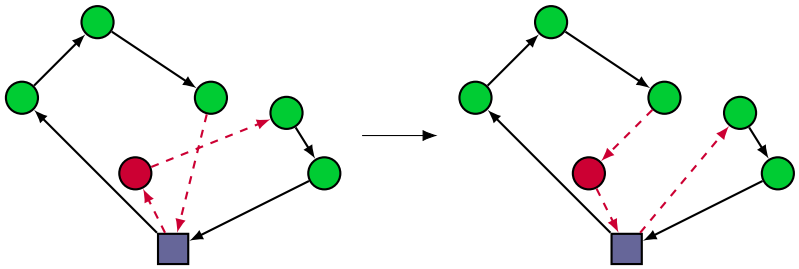


Figure 2.6: Example of an inter-route relocate move.

EXCHANGE NEIGHBORHOOD. The *exchange* neighborhood consists of the set $\Omega_{\text{exch}}(\mathbf{R}, \boldsymbol{\kappa})$ of all solutions that can be obtained by swapping the positions of two distinct customers in $\mathbf{R}$. If the two customers lie on the same route, then the move is known as an intra-route exchange move, whereas if they lie on a different route, then it is referred to as an inter-route exchange or *swap(1,1)* move, see Figure 2.6. The size of this neighborhood is also $\mathcal{O}(n^2)$.

2-OPT NEIGHBORHOOD. The *2-opt* neighborhood consists of the set $\Omega_{\text{2-opt}}(\mathbf{R}, \boldsymbol{\kappa})$ of all solutions that can be obtained by deleting two non-adjacent edges in $\mathbf{R}$ and replacing them with two other edges. If the two deleted edges lie on the same route, then the move is known as an intra-route 2-opt, whereas if they lie on a different route, then it is

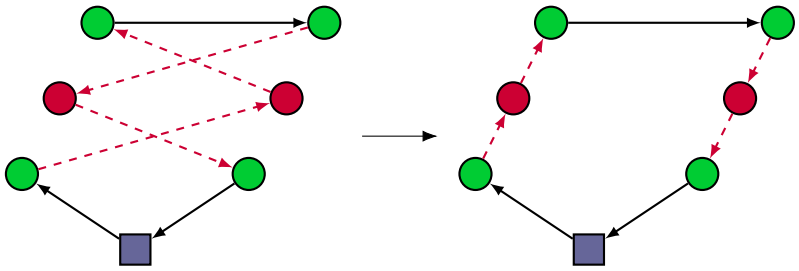


Figure 2.7: Example of an intra-route exchange move.

referred to as an inter-route 2-opt or *cross* or *2-opt** move, see Figure 2.6. The size of this neighborhood is also $\mathcal{O}(n^2)$.

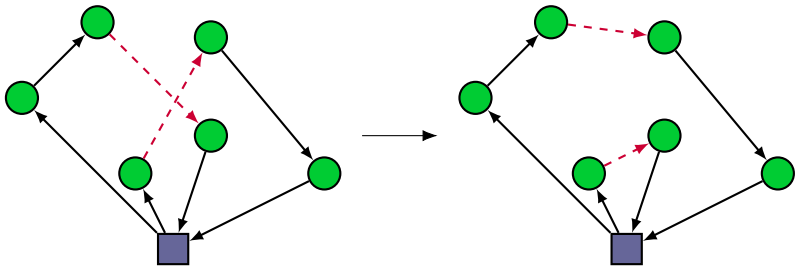


Figure 2.8: Example of an inter-route 2-opt move.

The goal of a local search algorithm is to efficiently explore the given set of neighborhoods to find *improving neighbor solutions*, *i.e.*, solutions $(\mathbf{R}', \boldsymbol{\kappa}')$ that are better than the current one $(\mathbf{R}, \boldsymbol{\kappa})$. To formalize what we mean by "better", we shall define the *total penalized cost* of a solution, $\bar{c}\,(\mathbf{R}, \boldsymbol{\kappa})$, as the weighted sum of its transportation costs and worst-case vehicle capacity violations:

$$\bar{c}\,(\mathbf{R}, \boldsymbol{\kappa}) = c\,(\mathbf{R}, \boldsymbol{\kappa}) + \varphi^{Q} p\,(\mathbf{R}, \boldsymbol{\kappa}) = c\,(\mathbf{R}, \boldsymbol{\kappa}) + \varphi^{Q} \sum_{h=1}^{H} \max\left\{0, \max_{q \in \mathcal{Q}} \sum_{i \in R_h} q_i - Q_{\kappa_h}\right\}, \quad (2.12)$$

where φ^{Q} is a large penalty coefficient. Therefore, a neighbor solution $(\mathbf{R}', \boldsymbol{\kappa}')$ is said to be improving if $\bar{c}\,(\mathbf{R}', \boldsymbol{\kappa}') < \bar{c}\,(\mathbf{R}, \boldsymbol{\kappa})$. By making φ^{Q} sufficiently large, this is equivalent to lexicographically comparing $p\,(\mathbf{R}', \boldsymbol{\kappa}') < p\,(\mathbf{R}, \boldsymbol{\kappa})$ and only if these are equal, then comparing if $c\,(\mathbf{R}', \boldsymbol{\kappa}') < c\,(\mathbf{R}, \boldsymbol{\kappa})$. We note that by doing this, we allow local search to handle also infeasible solutions. Specifically, if the initial solution to local search (*e.g.*, obtained via a construction heuristic) satisfies the visit constraints **(C1)** and fleet availability constraints **(C2)** but not necessarily the vehicle capacity constraints **(D3)**, then the local search algorithm will first attempt to find a robust feasible neighbor solution. Once such a neighbor solution is found, the local search does not reenter the infeasible region, thereafter admitting only those neighbor solutions that are feasible.

The above observations imply that even if we restrict our attention to the aforementioned neighborhoods, every iteration of local search involves evaluating the total penalized cost

of $\mathcal{O}(n^2)$ neighbor solutions to find at least one improving neighbor solution $(\mathbf{R}', \boldsymbol{\kappa}')$. While the evaluation of the transportation cost $c(\mathbf{R}', \boldsymbol{\kappa}')$ can be done in constant time (by simply updating the current value of $c(\mathbf{R}, \boldsymbol{\kappa})$), the evaluation of its robust feasibility, i.e., $p(\mathbf{R}', \boldsymbol{\kappa}')$, is not trivial, since it amounts to the solution of an inner optimization problem. As such, intra-route moves are trivial to evaluate since they only alter the positions of customers within a route, and not the actual set of customers itself, so that $p(\mathbf{R}', \boldsymbol{\kappa}') = p(\mathbf{R}, \boldsymbol{\kappa})$ in such cases. On the other hand, inter-route moves, which account for the majority of postulated moves, require us to recalculate $p(\mathbf{R}', \boldsymbol{\kappa}')$. Fortunately, whenever the uncertainty set $\mathcal{Q}$ is one of the five classes of sets described in Section 2.3.1, we can employ the data structures described in Proposition 2.2 to efficiently update the worst-case load of the affected routes (often in constant or sublinear time), and hence efficiently compute $p(\mathbf{R}', \boldsymbol{\kappa}')$. Specifically, since each of the aforementioned inter-route moves can be broken down into elementary additions and removals of customers, we can incrementally update the current $p(R, k)$ value of an affected route R to obtain the updated $p(R', k')$ value, using the result stated in Proposition 2.2.

In the context of the HVRP, efficient evaluation of solutions allows us to further generalize the aforementioned neighborhoods to modify also the fleet composition vector $\boldsymbol{\kappa}$ in addition to the routes $\mathbf{R}$. Specifically, whenever the fleet size is unlimited, every inter-route move also modifies the vehicle type κ_h of each affected route R_h. The goal is to minimize the worst-case vehicle capacity violation, i.e., $\kappa_h = \arg\min_{k \in K} p(R_h, k)$; if robust feasibility $(p(R_h, k) = 0)$ is possible using at least one vehicle type $k \in K$, then the goal is to minimize the transportation cost, i.e., $\kappa_h = \arg\min_{k \in K} \{c(R_h, k) : p(R_h, k) = 0\}$. In other words, the inter-route moves modify the vehicle type κ_h to the one that can feasibly perform route R_h under all realizations of the uncertainty at minimum cost. On the other hand, whenever the fleet size is limited, every inter-route move only exchanges the vehicle types of the affected routes (if doing so can lower $\bar{c}(\mathbf{R}, \boldsymbol{\kappa})$). This ensures that the fleet availability condition **(C2)** is never violated.

The above modifications apply to all neighborhoods. In addition to these, we can also generalize the definition of specific neighborhoods. For example, prior to exploring the relocate neighborhood, we can add an empty route to the current solution. The vehicle type of this empty route is one with the largest capacity for which it is possible to do so without violating condition **(C2)**. This allows the relocation of a customer to its own route, thus further expanding the search space.

Finally, we note that it is also possible to generalize the total penalized cost function $\bar{c}$ for specific HVRP variants (see Table 2.1). For example, to ensure that site dependencies are always respected in the SDVRP, we can generalize $\bar{c}$ to also include a penalty term for site violations:

$$\bar{c}^S(\mathbf{R}, \boldsymbol{\kappa}) = \bar{c}(\mathbf{R}, \boldsymbol{\kappa}) + \varphi^S s(\mathbf{R}, \boldsymbol{\kappa}) = \bar{c}(\mathbf{R}, \boldsymbol{\kappa}) + \varphi^S \sum_{h=1}^{H} \max\left\{0, \sum_{i \in R_h} \mathbb{I}[\kappa_h \notin K_i]\right\},$$

where φ^S is a large penalty coefficient. We note that the performance of the local search algorithm does not depend critically on the chosen values of the penalty coefficients φ^Q and φ^S as long as they are sufficiently large when compared to the transportation costs.

2.4.2 *Iterated Local Search*

Iterated Local Search (ILS), as the name implies, refers to the repeated application of local search to a current solution. The current solution may either be generated from scratch using a *construction heuristic* or by *perturbing* a locally optimal solution. We refer the reader to [191] for an introduction to this subject. Our specific ILS implementation for the robust HVRP is described in Algorithm 1.

Algorithm 1 Iterated local search.

Input: χ, η, ν, ζ, and δ (user-defined parameters)

1: Start timer t, $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right) \leftarrow (\emptyset, \emptyset)$
2: **while** $t < t_{\text{lim}}$ **do**
3: $\quad (\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ CONSTRUCT SOLUTION(η) ▷ Construction phase
4: $\quad (\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ TABU SEARCH$((\mathbf{R}, \boldsymbol{\kappa}), \nu, \zeta)$
5: $\quad$ **if** $\bar{c}(\mathbf{R}, \boldsymbol{\kappa}) < \bar{c}\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$ **then** $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right) \leftarrow (\mathbf{R}, \boldsymbol{\kappa})$ **end if**
6: $\quad$ counter $\leftarrow 0$, $\left(\mathbf{R}', \boldsymbol{\kappa}'\right) \leftarrow (\mathbf{R}, \boldsymbol{\kappa})$ ▷ Perturbation phase
7: $\quad$ **while** counter $< \chi$ **do**
8: $\quad\quad (\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ PERTURB SOLUTION$((\mathbf{R}, \boldsymbol{\kappa}), \delta)$
9: $\quad\quad (\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ TABU SEARCH$((\mathbf{R}, \boldsymbol{\kappa}), \nu, \zeta)$
10: $\quad\quad$ **if** $\bar{c}(\mathbf{R}, \boldsymbol{\kappa}) < \bar{c}\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$ **then** $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right) \leftarrow (\mathbf{R}, \boldsymbol{\kappa})$ **end if**
11: $\quad\quad$ **if** $\bar{c}(\mathbf{R}, \boldsymbol{\kappa}) < \bar{c}\left(\mathbf{R}', \boldsymbol{\kappa}'\right)$ **then**
12: $\quad\quad\quad$ counter $\leftarrow 0$
13: $\quad\quad\quad \left(\mathbf{R}', \boldsymbol{\kappa}'\right) \leftarrow (\mathbf{R}, \boldsymbol{\kappa})$
14: $\quad\quad$ **else**
15: $\quad\quad\quad$ counter $\leftarrow$ counter $+ 1$
16: $\quad\quad$ **end if**
17: $\quad$ **end while**
18: **end while**
19: **return** $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$

Algorithm 1 consists of two phases: a *construction* phase (lines 3–5) and a *perturbation* phase (lines 6–17). The construction phase first constructs an initial solution (line 3) and then improves it using an efficient local search algorithm called *tabu search* (line 4). The perturbation phase iteratively perturbs this solution (line 8) and improves it using tabu search (line 9). The perturbation phase terminates if it fails to encounter a solution that is better than the best one found in the current iteration $\left(\mathbf{R}', \boldsymbol{\kappa}'\right)$ for more than χ attempts (line 7). The ILS algorithm terminates after a pre-specified time limit t_{lim} is reached (line 2), at which point the best encountered solution $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$ is returned (line 19). The

parameters η, ν, ζ and δ are used as inputs to the construction heuristic, tabu search and perturbation mechanisms, respectively, which we describe next.

CONSTRUCTION HEURISTIC. The procedure CONSTRUCT SOLUTION(η) works by gradually inserting customers into an initially empty solution. At any given iteration, an empty route is first constructed for each vehicle type $k \in K$. To ensure that the fleet availability constraint **(C2)** is satisfied, this is done only if the number of routes of vehicle type k in the current partial solution is less than its available number m_k. Assuming this is the case, we keep adding unrouted customers to this route until it is no longer possible to do so (*i.e.*, either because all customers have been routed or because the capacity condition **(D3)** would be violated). Specifically, all customers that can be potentially added to the route are first inserted into a *restricted candidate list*; a random customer is then selected from this list and inserted into the position that greedily minimizes the corresponding *insertion cost*. In our implementation, the restricted candidate list is cardinality-based and fixed to a pre-defined size η. The parameter η determines the extent of randomization and greediness during the construction process. Based on empirical evidence, a value of $\eta \in [1, 10]$ appeared to work well in our numerical experiments.

If a route was successfully constructed for at least one vehicle type, we select the route R of vehicle type k for which the *average cost per unit of carried load*, defined as $c(R,k) / \max_{q \in Q} \sum_{i \in R} q_i$, is minimized. If no route could be constructed, then any remaining unrouted customer is inserted into an existing route (and corresponding position) for which a randomly weighted sum of the capacity violation and insertion cost is minimized. We remark that efficient computation of the average cost per unit of carried load is enabled by the data structures described in Proposition 2.2.

TABU SEARCH. In principle, we can replace all calls to the TABU SEARCH$((\mathbf{R}, \boldsymbol{\kappa}), \nu, \zeta)$ procedure by a simple local search algorithm that iteratively explores each pre-defined neighborhood to determine an improving solution. However, doing so will result in solutions that are only locally optimal with respect to the given neighborhoods, *i.e.*, all neighbor solutions $(\mathbf{R}', \boldsymbol{\kappa}') \in \Omega_Y(\mathbf{R}, \boldsymbol{\kappa})$ in each of the pre-defined neighborhoods Y are non-improving: $\bar{c}(\mathbf{R}', \boldsymbol{\kappa}') \geq c(\mathbf{R}, \boldsymbol{\kappa})$. Tabu search [129] overcomes this shortcoming of local search and enhances its performance in two important ways: *(i)* non-improving moves are allowed, and *(ii)* improving moves may be disallowed. This enhancement is achieved using a short term memory (also known as a *tabu list*) to keep track of the most recently visited solutions in the search history and to prevent revisiting them for a predefined number of local search iterations ν (the *tabu tenure*). Any potential solution that has been visited within the last ν iterations is marked "tabu" (or forbidden) and inserted into the tabu list, so that the algorithm does not cycle by repeatedly visiting the same solutions. In fact, an admissible neighbor solution that is in the tabu list can be visited only if certain *aspiration criteria* are met; specifically, the tabu status of a solution is overridden only if it improves upon the best encountered solution. Moreover, the tabu

search terminates if it performs ζ local search iterations without observing any further improvement. Typical values for these parameters are $\nu \in [20, 40]$ and $\zeta \in [100, 500]$ and in our implementation, we set $\nu = 30$ and $\zeta = 500$. Furthermore, we considered the set of neighborhoods to be the (intra- and inter-route) 1-0 relocate, 1-1 exchange and 2-opt neighborhoods. At each iteration, we randomly selected one of these neighborhoods and traversed it in lexicographic order, applying pruning mechanisms based on both feasibility and gain. The first improving neighbor solution replaced the current solution. Since the size of each of these neighborhoods is $\mathcal{O}(n^2)$, each iteration of tabu search itself can take $\mathcal{O}(n^2)$ time in the worst case, and its run time is largely dictated by the run time of the local search algorithm described in the previous section.

PERTURBATION MECHANISM. The procedure PERTURB SOLUTION$((\mathbf{R}, \boldsymbol{\kappa}), \delta)$ attempts to perturb the current solution $(\mathbf{R}, \boldsymbol{\kappa})$ such that the new solution cannot be encountered by application of tabu search alone. In our implementation, we consider a perturbation mechanism that first removes the route R of vehicle type k from the current solution which has the maximum value of *average cost per unit of carried load*. In addition to this route, the mechanism also removes any routes R' that are "sufficiently close" to R, *i.e.*, routes R' for which distance$(R, R') := \max_{(i,j)\in R\times R'} \underline{c}_{ij} < \delta$. Here, $\underline{c}_{ij}$ is any suitably defined distance measure between customers i and j (*e.g.*, geographical distance between i and j), and in our implementation, we set $\underline{c}_{ij} = \min_{k\in K} c_{ijk}$. If all routes R' satisfy distance$(R, R') \geq \delta$, then the route R'' which has the smallest distance to R is removed from the current solution. All customers that were visited on the deleted routes are then considered to be unrouted and are added back to the current partial solution using the same greedy insertion procedure that was used in the construction heuristic. We note that the parameter δ determines the extent of perturbation, with larger values of δ corresponding to higher extents of perturbation.

2.4.3 *Adaptive Memory Programming*

Adaptive Memory Programming (AMP) is a metaheuristic that focuses on the exploitation of strategic memory components. Based on the intuition that high-quality locally optimal solutions share common features (e.g., common customer visiting sequences), AMP attempts to exploit a set of long-term memories (in contrast to the short-term memory used in tabu search) for the iterative construction of new *provisional solutions*. These solutions are used to restart and intensify the search, while adaptive learning mechanisms are applied to update the memory structures. We refer the reader to [92, 129] for a general overview of this subject. Our specific AMP implementation for the robust HVRP is described in Algorithm 2.

Algorithm 2 consists of two phases: an *initialization* phase (lines 2–7) and an *exploitation* phase (lines 8–13). The initialization phase populates the *reference set* $\mathcal{P}$ with μ solutions that are generated by first constructing an initial solution (line 3) and then improving it

Algorithm 2 Adaptive Memory Programming.

Input: μ, η, ν, ζ, and θ (user-defined parameters)

1: Start timer t, $\mathcal{P} \leftarrow \emptyset$, $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right) \leftarrow (\emptyset, \emptyset)$
2: **while** $|\mathcal{P}| < \mu$ **do** ▷ Initialization phase
3: $(\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ CONSTRUCT SOLUTION(η)
4: $(\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ TABU SEARCH($(\mathbf{R}, \boldsymbol{\kappa}), \nu, \zeta$)
5: **if** $\bar{c}(\mathbf{R}, \boldsymbol{\kappa}) < \bar{c}\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$ **then** $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right) \leftarrow (\mathbf{R}, \boldsymbol{\kappa})$ **end if**
6: $\mathcal{P} \leftarrow \mathcal{P} \cup (\mathbf{R}, \boldsymbol{\kappa})$
7: **end while**
8: **while** $t < t_{\text{lim}}$ **do** ▷ Exploitation phase
9: $(\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ CONSTRUCT PROVISIONAL SOLUTION($\mathcal{P}, \theta$)
10: $(\mathbf{R}, \boldsymbol{\kappa}) \leftarrow$ TABU SEARCH($(\mathbf{R}, \boldsymbol{\kappa}), \nu, \zeta$)
11: **if** $\bar{c}(\mathbf{R}, \boldsymbol{\kappa}) < \bar{c}\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$ **then** $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right) \leftarrow (\mathbf{R}, \boldsymbol{\kappa})$ **end if**
12: $\mathcal{P} \leftarrow$ UPDATE REFERENCE SET($\mathcal{P}, (\mathbf{R}, \boldsymbol{\kappa})$)
13: **end while**
14: **return** $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$

using tabu search (line 4). Once the initialization phase has completed, the exploitation phase manipulates $\mathcal{P}$ by exploring search trajectories initiated using the provisional solutions as starting points. Specifically, at each iteration of the exploitation phase, a provisional solution is first constructed by identifying common features of the reference solutions in $\mathcal{P}$ (line 9). This provisional solution is then further improved using tabu search (line 10) and inserted into the reference set $\mathcal{P}$ (line 12). The AMP algorithm terminates after a pre-specified time limit t_{lim} is reached (line 8), at which point the best encountered solution $\left(\mathbf{R}^B, \boldsymbol{\kappa}^B\right)$ is returned (line 14). The procedures CONSTRUCT SOLUTION(η) and TABU SEARCH($(\mathbf{R}, \boldsymbol{\kappa}), \nu, \zeta$) along with their associated parameters η, ν and ζ are exactly the same as in ILS (see Algorithm 1). The parameters θ and μ are used in the provisional construction and reference set update methods respectively, which we describe next.

GENERATION OF PROVISIONAL SOLUTIONS. Provisional solutions are constructed by identifying and combining *elite components* from the reference set $\mathcal{P}$. We define an elite component to be a route associated with a particular vehicle type whose edges appear "sufficiently frequently" among the solutions in $\mathcal{P}$. The overall procedure CONSTRUCT PROVISIONAL SOLUTION($\mathcal{P}, \theta$) works as follows. We first attempt to generate a route for every vehicle type $k \in K$, assuming the number of routes of this type is less than m_k in the current solution. With probability θ, we construct a route using the members of $\mathcal{P}$, and with probability $1 - \theta$, we use the mechanism outlined in the basic CONSTRUCT SOLUTION(η) procedure. In the former case, we first assign a score to each route $R_h = (r_{h1}, \ldots, r_{h|R|})$ of vehicle type κ_h from the solution $(\mathbf{R}, \boldsymbol{\kappa}) \in \mathcal{P}$ as follows: $\text{score}(R_h, \kappa_h) = \text{w}(\mathbf{R}, \boldsymbol{\kappa}) \sum_{l=0}^{|R|} \text{freq}(r_{hl}, r_{hl+1}, k) \mathbb{I}[r_{hl}, r_{hl+1} \notin \text{current solution}]$. Here $\text{w}(\mathbf{R}, \boldsymbol{\kappa})$ refers to the

weight of solution $(\mathbf{R}, \boldsymbol{\kappa})$ while $\text{freq}(i, j, k)$ refers to the frequency with which edge $(i, j) \in E$ appears among all routes of vehicle type k. Specifically, these quantities are defined as follows:

$$\text{w}(\mathbf{R}, \boldsymbol{\kappa}) = \frac{\max_{(\mathbf{R}', \boldsymbol{\kappa}') \in \mathcal{P}} \bar{c}(\mathbf{R}', \boldsymbol{\kappa}') - \bar{c}(\mathbf{R}, \boldsymbol{\kappa})}{\max_{(\mathbf{R}', \boldsymbol{\kappa}') \in \mathcal{P}} \bar{c}(\mathbf{R}', \boldsymbol{\kappa}') - \min_{(\mathbf{R}', \boldsymbol{\kappa}') \in \mathcal{P}} \bar{c}(\mathbf{R}', \boldsymbol{\kappa}')},$$

$$\text{freq}(i, j, k) = \sum_{(\mathbf{R}', \boldsymbol{\kappa}') \in \mathcal{P}} \sum_{h=1}^{H'} \mathbb{I}[\kappa'_h = k] \sum_{l=0}^{|R'_h|} \mathbb{I}[(r'_{hl}, r'_{hl+1}) = (i, j)].$$

The route R with the highest score is then determined to be the candidate route of vehicle type k (after deleting those customers that have already been routed in the current solution).

Among all generated routes, the candidate route of the vehicle type with the lowest value of *average cost per unit of carried load* is then adopted in the current solution, and the entire procedure repeats. At the end, if unrouted customers still remain, then they are inserted into an existing route (and corresponding position) for which a randomly weighted sum of the capacity violation and insertion cost is minimized, similar to the basic construction heuristic. We recall that this is done so that the fleet availability constraints **(C2)** are never violated.

REFERENCE SET UPDATE METHOD. In the initialization phase, the reference set $\mathcal{P}$ is grown to contain up to μ different solutions. In the exploitation phase, the size of $\mathcal{P}$ is kept constant by replacing older solutions with more recently encountered ones. To ensure an appropriate balance between quality and diversity among the reference solutions, the procedure UPDATE REFERENCE SET$(\mathcal{P}, (\mathbf{R}, \boldsymbol{\kappa}))$ uses a simple rule that replaces the worst solution (in terms of its total cost) $(\mathbf{R}^W, \boldsymbol{\kappa}^W)$ whenever the candidate solution to be inserted $(\mathbf{R}, \boldsymbol{\kappa})$ satisfies $\bar{c}(\mathbf{R}, \boldsymbol{\kappa}) < \bar{c}(\mathbf{R}^W, \boldsymbol{\kappa}^W)$.

2.5 ROBUST INTEGER PROGRAMMING MODEL AND BRANCH-AND-CUT

The metaheuristic approaches described in the previous section determine high-quality robust feasible solutions, in general. To precisely quantify the quality of these solutions however, we need a lower bound on the optimal objective value of the robust HVRP. To that end, Section 2.5.1 describes an integer programming (IP) formulation whose optimal solution coincides with that of the robust HVRP, while Section 2.5.2 describes a branch-and-cut algorithm for its solution. A lower bound on the optimal objective value can then be obtained by recording the global lower bound of the associated branch-and-bound tree after a given amount of time.

2.5.1 *Integer Programming Model*

Our model is similar to the classical *vehicle-flow formulation* originally introduced in [180] for the CVRP. The model uses binary variables y_{ik} to record if customer $i \in V_C$ is visited by a vehicle of type $k \in K$ and variables x_{ijk} to record if the edge $(i,j) \in E$ is traversed by a vehicle of type $k \in K$. To facilitate the description of the formulation, we define $V_k := \{i \in V_C : \max_{q \in \mathcal{Q}} q_i \leq Q_k\}$ to be the subset of those customers that can be visited by a vehicle of type $k \in K$ under any customer demand realization $q \in \mathcal{Q}$. For a given $S \subseteq V_k$, we also define $\delta_k(S)$ to be the subset of all edges in E with one end point in S and the other in $V_k \setminus S$. The complete formulation is as follows.

$$\underset{x,y}{\text{minimize}} \quad \sum_{k \in K} \sum_{i \in V_C : (0,i) \in E} (f_k/2) x_{0ik} + \sum_{k \in K} \sum_{(i,j) \in E} c_{ijk} x_{ijk} \tag{2.13a}$$

$$\text{subject to} \quad y_{ik} \in \{0,1\} \quad \forall i \in V_C,\ \forall k \in K, \tag{2.13b}$$

$$x_{ijk} \in \{0,1\} \quad \forall (i,j) \in E \cap (V_C \times V_C),\ \forall k \in K, \tag{2.13c}$$

$$x_{0ik} \in \{0,1,2\} \quad \forall i \in V_C : (0,i) \in E,\ \forall k \in K, \tag{2.13d}$$

$$\sum_{k \in K} y_{ik} = \sum_{k \in K : i \in V_k} y_{ik} = 1 \quad \forall i \in V_C, \tag{2.13e}$$

$$\sum_{j \in V : (i,j) \in E} x_{ijk} = 2 y_{ik} \quad \forall i \in V_C,\ \forall k \in K, \tag{2.13f}$$

$$\sum_{i \in V_C : (0,i) \in E} x_{0ik} \leq 2 m_k \quad \forall k \in K, \tag{2.13g}$$

$$\sum_{(i,j) \in \delta_k(S)} x_{ijk} + 2 \sum_{i \in S} (1 - y_{ik}) \geq 2 \left\lceil \frac{1}{Q_k} \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i \right\rceil \quad \forall S \subseteq V_k,\ \forall k \in K. \tag{2.13h}$$

The objective function (2.13a) minimizes the sum of the fixed costs and transportation costs; since the fixed costs are accounted on both the first and last edges of a route (*i.e.*, both edges adjacent to the depot), the total must be divided by two to obtain the true fixed cost. Constraints (2.13b)–(2.13d) enforce integrality restrictions. Constraints (2.13e) stipulate that each customer must be visited by exactly one vehicle type; moreover, it also requires that this vehicle type be such that it can feasibly visit this customer under all possible demand realizations. Constraints (2.13f) enforce that if a customer is visited by vehicle type $k \in K$, then there are exactly two edges adjacent to it that are traversed by a vehicle of that type; moreover, if it is not visited by a vehicle of type $k \in K$, then all edge variables adjacent to it are set to zero. Constraints (2.13g) ensure that no more than m_k vehicles of type $k \in K$ are used. Constraints (2.13h) restrict *subtours* (*i.e.*, ensure that no vehicles of type $k \in K$ perform routes that are disconnected from the depot) and also enforce that the worst-case load of routes of type $k \in K$ are less than the capacity Q_k. We refer to these inequalities as *robust heterogeneous rounded capacity inequalities* since they generalize the classical *rounded capacity inequalities* (RCI) for the deterministic CVRP [193] (*i.e.*, whenever $\mathcal{Q}$ and K are both singletons). Finally, we note

that in the case of the SDVRP (see Table 2.1), the following additional constraint can be added to formulation (2.13) to ensure that all site dependencies are respected:

$$y_{ik} = 0 \quad \forall k \notin K_i,\ \forall i \in V_C. \tag{2.13i}$$

Proposition 2.3 establishes the correctness of formulation (2.13).

Proposition 2.3. *The feasible solutions of formulation* (2.13) *are in one-to-one correspondence with robust feasible solutions of the HVRP.*

Proof. Suppose $(\mathbf{R}, \boldsymbol{\kappa})$ is a robust feasible solution of the HVRP, *i.e.*, it satisfies conditions **(C1)**, **(C2)** and **(D3)**. Construct the solution (x, y) as follows: $y_{ik} = \sum_{h=1}^{H} \mathbb{I}[i \in R_h]\mathbb{I}[k = \kappa_h]$ and $x_{ijk} = \sum_{h=1}^{H} \sum_{l=0}^{|R_h|} \mathbb{I}[(i,j) = (r_{hl}, r_{hl+1})]\mathbb{I}[k = \kappa_h]$. We claim that (x, y) is a feasible solution of formulation (2.13). To see this, first observe that satisfaction of constraints (2.13b)–(2.13f) follows from the definition of a route (see Section 2.2) and from the fact that $(\mathbf{R}, \boldsymbol{\kappa})$ satisfies condition **(C1)**. Similarly, constraint (2.13g) is satisfied because $(\mathbf{R}, \boldsymbol{\kappa})$ satisfies condition **(C2)**. Constraints (2.13h) are satisfied because of the following reason. First, observe that we have:

$$\begin{aligned}
\sum_{(i,j)\in\delta_k(S)} x_{ijk} &\geq 2 \Big| \underbrace{h \in \{1,\ldots,H\} : \kappa_h = k \text{ and } S \cap R_h \neq \emptyset}_{:= H_k(S) = \text{index set of routes of type } k \text{ 'crossing' } S} \Big| \\
&= 2 \left\lceil \frac{1}{Q_k} \sum_{h \in H_k(S)} Q_k \right\rceil \\
&\geq 2 \left\lceil \frac{1}{Q_k} \max_{q \in \mathcal{Q}} \sum_{h \in H_k(S)} \sum_{i \in S \cap R_h} q_i \right\rceil = 2 \left\lceil \frac{1}{Q_k} \max_{q \in \mathcal{Q}} \sum_{i \in S \cap \left(\cup_{h \in H_k(S)} R_h\right)} q_i \right\rceil,
\end{aligned}$$

where the first inequality follows by construction of x while the second inequality follows because *(i)* $(\mathbf{R}, \boldsymbol{\kappa})$ satisfies condition **(D3)** and *(ii)* the maximum operator is subadditive. Second, we have:

$$2 \sum_{i \in S} (1 - y_{ik}) = 2 \left| i \in S \setminus \left(\cup_{h \in H_k(S)} R_h \right) \right| \geq 2 \left\lceil \frac{1}{Q_k} \max_{q \in \mathcal{Q}} \sum_{i \in S \setminus \left(\cup_{h \in H_k(S)} R_h\right)} q_i \right\rceil,$$

where the equality follows by construction of y while the inequality follows because *(i)* each $i \in S \subseteq V_k$ satisfies $\max_{q \in \mathcal{Q}} q_i \leq Q_k$ and, *(ii)* the maximum and ceiling operators are subadditive. Finally, combining the above two expressions and using again the subadditivity of the maximum and ceiling operators shows that inequalities (2.13h) are satisfied.

Now, suppose that (x, y) is a feasible solution of formulation (2.13). Construct $(\mathbf{R}, \boldsymbol{\kappa})$ as follows: *(i)* $H \leftarrow 0$; *(ii)* for every $i \in V_C$, if $\sum_{k \in K} x_{0ik} = 1$ and $i \notin R_1, \ldots, R_H$, then set $H \leftarrow H + 1$ and define $\kappa_H = \sum_{k \in K} k\mathbb{I}[y_{ik} = 1]$ and R_H to be the cycle that passes through customer i in the subgraph of G induced by $\{(i', j') \in E : x_{i'j'\kappa_H} = 1\}$. We claim

that $(\mathbf{R}, \boldsymbol{\kappa})$ is a robust feasible solution of the HVRP. To see this, first observe that $(\mathbf{R}, \boldsymbol{\kappa})$ satisfies condition **(C1)** because of inequalities (2.13b)–(2.13f), and condition **(C2)** because of inequality (2.13g). To see that condition **(D3)** is also satisfied for each route R_h, $h \in \{1, \ldots, H\}$: set $S = R_h$ and $k = \kappa_h$ in inequalities (2.13h). The left-hand side simplifies to 2 while the right-hand side simplifies to $2\lceil (1/Q_{\kappa_h}) \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i \rceil$. Hence, we have $1 \geq \lceil (1/Q_{\kappa_h}) \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i \rceil$, which implies that condition **(D3)** is satisfied. □

2.5.2 *Branch-and-Cut Algorithm*

The number of variables in formulation (2.13) is $\mathcal{O}(mn^2)$ but the number of constraints is $\mathcal{O}(m2^n)$. If we leave out the RCI constraints (2.13h) however, then the number of remaining constraints is $\mathcal{O}(mn)$. Therefore, we can solve the formulation in a *cutting plane* fashion by removing constraints (2.13h) and dynamically re-introducing them if they are found to be violated by the solution of the current linear programming relaxation. In fact, we can embed the cutting plane generation in each node of a branch-and-bound tree to obtain a *branch-and-cut* algorithm. We refer to [193] for a general reference on branch-and-cut in the context of vehicle routing. The performance of the branch-and-cut algorithm can be improved by adding in each tree node, inequalities that are valid but not necessary for the correctness of formulation (2.13). In the following, we describe several such valid inequalities as well as the associated *separation algorithms*. We also describe an effective preprocessing step that can reduce the number of variables in formulation (2.13).

VALID INEQUALITIES. Several valid inequalities have been proposed for flow-based formulations of the deterministic HVRP in [27, 273]. Among these, the *cover inequalities* and *fleet-dependent capacity inequalities* are particularly effective in obtaining strong lower bounds. The validity of the inequalities for the robust HVRP formulation (2.13) can be established by defining them for every possible customer demand realization $q \in \mathcal{Q}$. To describe these inequalities, we assume, without loss of generality, that the vehicle types are sorted in increasing order with respect to their capacities: $Q_1 \leq \ldots \leq Q_m$. We also define $s_k(q)$ to be total demand of the customers for which a vehicle of type k is the smallest one that can visit them, under a particular customer demand realization $q \in \mathcal{Q}$. Then, the following *robust cover inequalities* are valid for formulation (2.13).

$$\sum_{h=k}^{m} \left(\lfloor \alpha Q_h \rfloor + \min \left\{ 1, \frac{\alpha Q_h - \lfloor \alpha Q_h \rfloor}{\alpha \sum_{h=k}^{m} s_h(q) - \left\lfloor \alpha \sum_{h=k}^{m} s_h(q) \right\rfloor} \right\} \right) \sum_{i \in V_C : (0,i) \in E} x_{0ih} \geq 2 \left\lceil \alpha \sum_{h=k}^{m} s_h(q) \right\rceil$$

$$\forall \alpha \in \mathbb{R}_+, \ \forall k \in K, \ \forall q \in \mathcal{Q}. \tag{2.13j}$$

Let us define $\delta(S)$ to be the set of edges in E with exactly one end point in S and one end point in $V_C \setminus S$. Then, the following *robust fleet-dependent capacity inequalities* are valid for formulation (2.13).

$$\sum_{k\in K}\sum_{(i,j)\in\delta(S)} x_{ijk} + \sum_{h=k}^{m}\left\lceil 2\left(\frac{Q_h - Q_{k-1}}{Q_{k-1}}\right)\right\rceil \sum_{i\in V_C:(0,i)\in E} x_{0ih} \geq \left\lceil \frac{2}{Q_{k-1}} \max_{q\in\mathcal{Q}} \sum_{i\in S} q_i \right\rceil \quad \forall S \subseteq V_C,\ \forall k \in K \setminus \{1\}. \tag{2.13k}$$

The validity of the following *generalized subtour elimination constraints* and *generalized fractional capacity inequalities* can also be easily verified. The term *generalized* refers to the fact that these inequalities reduce to the classical subtour elimination constraints and fractional capacity inequalities, respectively, in the case of the deterministic CVRP (*i.e.*, when both $\mathcal{Q}$ and K are singletons). However, unlike the latter, these inequalities do not dominate and are not dominated by the robust heterogeneous RCI constraints (2.13h).

$$\sum_{(i,j)\in\delta_k(S)} x_{ijk} \geq 2 \max_{v\in S} y_{vk} \quad \forall S \subseteq V_k,\ \forall k \in K. \tag{2.13l}$$

$$\sum_{(i,j)\in\delta_k(S)} x_{ijk} \geq \frac{2}{Q_k} \sum_{i\in S} q_i y_{ik} \quad \forall S \subseteq V_k,\ \forall k \in K,\ \forall q \in \mathcal{Q}. \tag{2.13m}$$

In addition to the above, any valid inequality for the *two-index vehicle flow formulation* of the deterministic CVRP that is defined over graph $G = (V, E)$ with vehicle capacity $Q = \max_{k\in K} Q_k$, can also be made valid for formulation (2.13) by defining it for all $q \in \mathcal{Q}$ and by replacing the two-index variable x_{ij} with $\sum_{k\in K} x_{ijk}$ for all $(i, j) \in E$. In our implementation, we used the comb, framed capacity and multistar inequalities in this manner [193].

SEPARATION ALGORITHMS. Let $(\bar{x}, \bar{y})$ be a fractional solution encountered in some node of the search tree. The goal of a separation algorithm is to identify if a particular member of a family of inequalities is violated by the current solution $(\bar{x}, \bar{y})$. Consider the robust heterogeneous RCI constraints (2.13h). For a particular vehicle type $k \in K$, the identification of a customer set $S \subseteq V_k$ for which the corresponding inequality is violated by $(\bar{x}, \bar{y})$ is nontrivial. In the deterministic CVRP, this is typically achieved by a local search algorithm. For example, a greedy search algorithm is presented in [193]; this algorithm iteratively expands a (randomly initialized) set $S = \{s\}$ with a customer j for which the corresponding slack of the RCI constraint (*i.e.*, difference between the right-hand side and left-hand side) is maximized. In our implementation, we extend this idea by using a tabu search procedure very similar to the one presented in Section 2.4. The key difference is that a "solution" in the context of this local search algorithm is simply a customer set $S \subseteq V_k$ as opposed to an entire set of routes. Specifically, the algorithm starts with a randomly selected customer set $S \subseteq V_k$ and then iteratively perturbs this set through a sequence of operations in which individual customers are added or removed. In each iteration, the algorithm greedily chooses a customer whose

inclusion or removal maximizes the slack of the corresponding robust heterogeneous RCI constraint (2.13h). Similar to the argument in Section 2.4, computing this slack requires the computation of the right-hand side which, in turn, requires the efficient computation of the worst-case load over the current candidate set of customers S. This is achieved by using the data structures described in Proposition 2.2. Finally, similar to Section 2.4, the algorithm also maintains tabu lists of customers that have recently been added or removed to avoid cycling and to escape local optima. The algorithm terminates if we cannot maximize the slack of constraint (2.13h) for more than a pre-defined number of consecutive iterations.

The separation algorithm for the robust fleet-dependent capacity inequalities (2.13k) is exactly the same as above. The separation problem for the generalized subtour elimination constraints (2.13l) is solved using the polynomial-time algorithm described in [113]. Similarly, the separation problem for the generalized fractional capacity inequalities (2.13m), under a particular demand realization $q^\star \in \mathcal{Q}$, reduces to the separation problem of the fractional capacity inequalities for the deterministic CVRP if we define the customer demands to be $q_i^\star \bar{y}_{ik}$, which is known to be polynomial-time solvable [197]. In our implementation, we restrict the separation to a particular realization defined by $q^\star \in \arg\max_{q \in \mathcal{Q}} \sum_{i \in V_C} q_i^\star$. Similarly, we separate the CVRP-based comb, framed capacity and multistar inequalities using the CVRPSEP package [193] by only considering $q^\star \in \mathcal{Q}$. Finally, the robust cover inequalities (2.13j) are separated by enumerating $\alpha \in \{Q_1, \ldots, Q_m, \gcd(Q_1, \ldots, Q_m)\}$, where gcd denotes the greatest common divisor, and by considering only $q^\star \in \mathcal{Q}$.

PREPROCESSING. A simple method to reduce the number of vehicle types was presented in [81] for the deterministic HVRP. Suppose UB is a known upper bound on the optimal objective value of formulation (2.13). For example, UB may obtained using the metaheuristics described in Section 2.4. Suppose we enforce now that at least one vehicle of type $k \in K$ must be used, by adding the constraint $\sum_{i \in V_C} y_{ik} \geq 1$ to formulation (2.13). If LB'_k denotes a lower bound on the optimal value of this augmented problem and if $LB'_k > UB$, then we can delete vehicle type $k \in K$ and all of its associated variables from formulation (2.13), since the corresponding solution would be suboptimal. In our implementation, we estimate LB'_k by solving the augmented formulation using a branch-and-cut algorithm and recording the global lower bound of the branch-and-bound tree after 1 minute.

2.6 COMPUTATIONAL RESULTS

This section presents computational results obtained using the metaheuristic algorithms described in Section 2.4 as well as the exact algorithm described in Section 2.5. Specifically, Section 2.6.1 provides an overview of the benchmark instances used; Section 2.6.2 presents a detailed computational study using the ILS and AMP algorithms; Section 2.6.3

illustrates the quality of the lower bounds obtained using the branch-and-cut algorithm; and finally, in Section 2.6.4, we analyze the robust HVRP solutions in terms of their robustness and objective value.

All algorithms were coded in C++ and compiled using the GCC 7.3.0 compiler. Each run was conducted on a single thread of an Intel Xeon 3.1 GHz processor. In the implementation of the ILS and AMP algorithms, the following parameter values were used: $\varphi^Q = 1000\underline{c}Q_{\max}^{-1}$, $\varphi^S = 100\underline{c}$, $\chi = 10$, $\eta = 3$, $\nu = 30$, $\eta = 500$, $\delta = 0.5\underline{c}$, $\theta = 0.7$ and $\mu = 16$, where we have defined $Q_{\max} = \max_{k \in K} Q_k$ and $\underline{c}_{ij} = \max_{(i,j) \in V_C \times V_C} \min_{k \in K} c_{ijk}$. An overall time limit of 1,000 seconds ($t_{\lim} = 1000$) was used. In the implementation of the branch-and-cut algorithm, we used CPLEX 12.7 as the IP solver; all solver options were at their default values with three exceptions: *(i)* general-purpose cutting planes and upper bounding heuristics were disabled, *(ii)* strong branching was enabled, and *(iii)* all valid inequalities described in Section 2.5.2 were added using user-defined callback functions. An overall time limit of 10,000 seconds was used in this case.

2.6.1 *Test Instances*

All our instances are based on the following three benchmark datasets corresponding to different variants of the deterministic HVRP.

(a) HVRP instances: We consider the twelve instances involving up to 100 customers proposed in [134] and adapted by [81, 246]. The data of these instances can be found at http://mistic.heig-vd.ch/taillard/problemes.dir/vrp.dir/vrp. The instances for the different HVRP variants are obtained by changing the data of the HVRPFD instances as follows, resulting in a total of 52 instances.

- HVRPD: Set $f_k = 0$ for each $k \in K$.
- FSMFD: Set $m_k = n$ for each $k \in K$.
- FSMD: Set $f_k = 0$ and $m_k = n$ for each $k \in K$.
- FSMF: Set $m_k = n$ for each $k \in K$ and $c_{ijk} = e_{ij}$, where e_{ij} is the Euclidean distance between nodes $i \in V$ and $j \in V$.

(b) SDVRP instances: We consider the 13 instances containing up to 108 customers that have also been considered by [29, 77, 88, 204]. The data of these instances can be found at http://neumann.hec.ca/chairedistributique/data/sdvrp.

(c) MDVRP instances: We consider the 9 instances involving up to 160 customers that have also been considered by [29, 87]. The data of these instances can be found at http://neumann.hec.ca/chairedistributique/data/mdvrp.

For each deterministic HVRP benchmark, we construct five classes of uncertainty sets. To ensure that the constructed sets are meaningful, we partition the customer set V_C into four geographic quadrants – NE, NW, SW, SE – based on the coordinates in the

benchmark instance. Moreover, the customer demands specified in the benchmark are taken to be their nominal values q^0. We then construct the following uncertainty sets, each of which is parametrized by scalars $\alpha, \beta \in [0,1]$.

(a) Budget sets (originally proposed in [137]):

$$\mathcal{Q}_B = \left\{ q \in [(1-\alpha)q^0, (1+\alpha)q^0] : \sum_{i\in\Omega} q_i \leq (1+\alpha\beta) \sum_{i\in\Omega} q_i^0 \quad \forall \Omega \in \{NE, NW, SW, SE\}. \right\}.$$

This set stipulates that each customer demand can deviate by at most $\alpha \cdot 100\%$ from its nominal value, but the cumulative demand in each quadrant may not exceed its nominal value by more than $\beta \cdot 100\%$.

(b) Factor models (originally proposed in [137]):

$$\mathcal{Q}_F = \left\{ q \in \mathbb{R}^n : q = q^0 + \Psi\xi \text{ for some } \xi \in \Xi_F \right\}, \text{where } \Xi_F = \left\{ \xi \in [-1,1]^4 : \left| e^\top \xi \right| \leq 4\beta \right\}.$$

This set models the demand of customer i as a convex combination of 4 factors that can be interpreted as quadrant demands with the weights reflecting the relative proximity of customer i to the quadrant. Specifically, we set $\Psi_{if} = \alpha q_i^0 \psi_{if} / \sum_{f'=1}^4 \psi'_{if}$, where ψ_{if} denotes the inverse distance between customer i and the centroid of quadrant $f \in \{1,2,3,4\}$.

(c) Ellipsoids:

$$\mathcal{Q}_E = \left\{ q \in \mathbb{R}^n : q = q^0 + \Sigma^{1/2}\xi \text{ for some } \xi \in \Xi_E \right\}, \text{where } \Xi_E = \left\{ \xi \in \mathbb{R}^n : \xi^\top \xi \leq 1 \right\}.$$

We define $\Sigma = (1-\beta)\Psi\Psi^\top + \beta \text{diag}\left(\alpha q_1^0, \ldots, \alpha q_n^0\right)^2$, where Ψ is the factor loading matrix defined above while $\text{diag}(\cdot)$ is a square diagonal matrix with $(\cdot)$ denoting the entries along its main diagonal. When $\beta = 0$, $\mathcal{Q}_E$ is approximated as a 4-dimensional ellipsoid centered at q^0 and the columns of Ψ represent the directions along its semi-axes. When $\beta \in (0,1)$, $\mathcal{Q}_E$ is a general n-dimensional ellipsoid centered at q^0. When $\beta = 1$, $\mathcal{Q}_E$ is an axis-parallel ellipsoid centered at q_0 with a semi-axis length of αq_i^0 along the i^{th} dimension; that is, when $\beta = 1$, $\mathcal{Q}_E$ inscribes the n-dimensional hyper-rectangle $[(1-\alpha)q^0, (1+\alpha)q^0]$.

(d) Cardinality-constrained sets:

$$\mathcal{Q}_G = \left\{ q \in [q^0, (1+\alpha)q^0] : q = q^0 + \alpha(q^0 \circ \xi) \text{ for } \xi \in \Xi_G \right\},$$
$$\text{where } \Xi_G = \left\{ \xi \in [0,1]^n : e^\top \xi \leq \beta n \right\}.$$

The set stipulates that each demand can deviate from its nominal value by up to $\alpha \cdot 100\%$ but the total number of customer demands that can simultaneously deviate is at most $\lceil \beta n \rceil$.

(e) Discrete sets:

$$\mathcal{Q}_D = \text{conv}\left(\left\{ q^0 \right\} \cup \left\{ q^{(j)} : j = 1, \ldots, \text{nint}(\beta n) \right\} \right).$$

Here, nint(βn) denotes the nearest integer to βn. The points $q^{(j)}$ are generated by uniformly sampling nint(βn) points from the n-dimensional hyper-rectangle $[(1-\alpha)q^0, (1+\alpha)q^0]$. Thus, the set approximates the customer demands as independent, uniform random variables.

The deterministic HVRP benchmarks are characterized by a high vehicle utilization under the nominal demands q^0; that is, the unused capacity of the vehicles is small, particularly in the case of problem variants with limited fleets (see Table 2.1). If the fleet size and vehicle capacities are unchanged, then several benchmark instances become infeasible in the presence of demand uncertainty. To alleviate this problem and conduct a meaningful computational study, we increase the capacity of each vehicle type Q_k in each benchmark by 10% (unless explicitly stated otherwise), which suffices to guarantee robust feasibility for $\alpha \leq 0.1$.

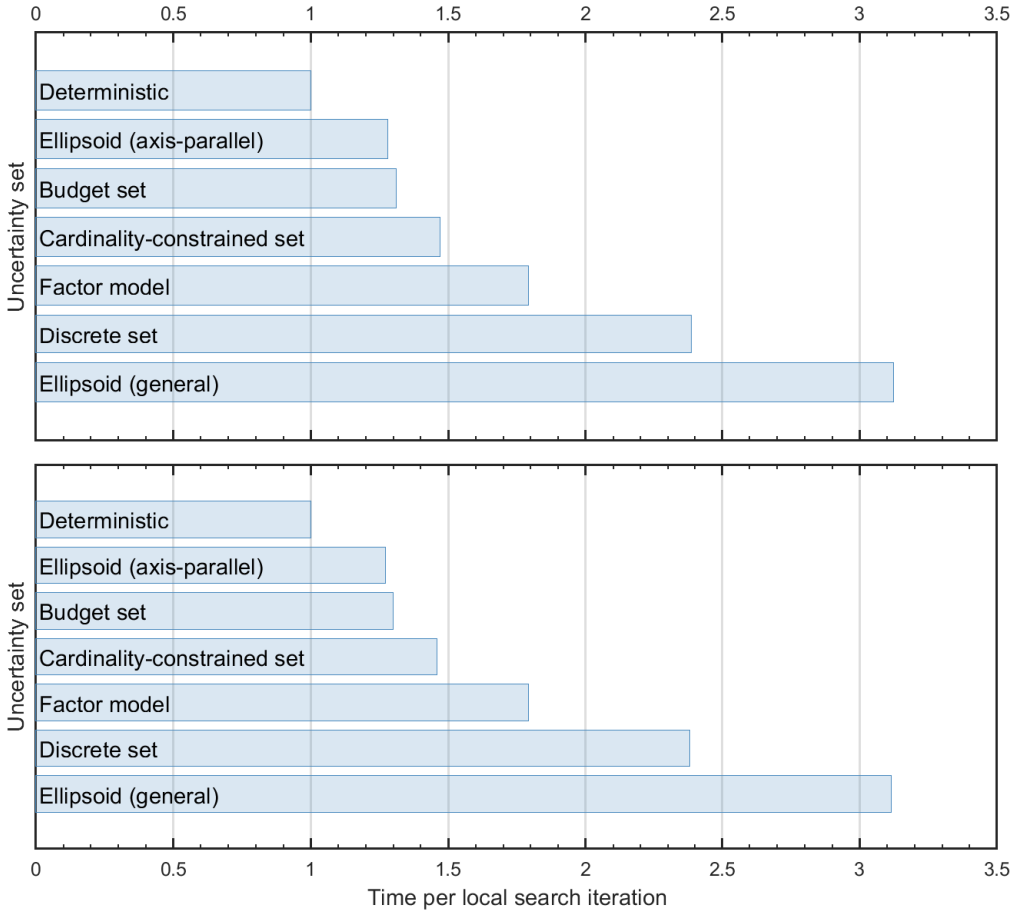
2.6.2 *Performance of Robust Local Search and Metaheuristics*

The results reported in this section are averages across 10 runs for each of the 74 test instances. For the budget sets, factor models and general ellipsoids, we set $(\alpha, \beta) = (0.1, 0.5)$, while for the cardinality-constrained and discrete sets, we set $(\alpha, \beta) = (0.1, 0.2)$. We note that the axis-parallel ellipsoid is obtained by setting $(\alpha, \beta) = (0.1, 1)$.

Figure 2.9 shows the time per local search iteration for the different classes of uncertainty sets described in Section 2.3. We note here that each iteration of local search involves evaluating $\mathcal{O}(n^2)$ neighbor solutions, see Section 2.4.1. We make the following observations from Figure 2.9. First, the time per iteration correlates well with the time complexities described in Table 2.3. Indeed, each local search iteration can be performed much faster when the uncertainty set is a budget set or axis-parallel ellipsoid since the worst-case load can be computed in constant time in such cases. In contrast, the local search iterations are relatively slower when the uncertainty set is a discrete set or general ellipsoid for which the worst-case load can only be computed in linear time (linear in $D = \beta n$ and n respectively). Second, the results are remarkably similar across the ILS and AMP algorithms. This shows that the time per iteration is dictated by the chosen uncertainty set and not by the overarching metaheuristic algorithm.

We note that the time limit of 1000 seconds is quite generous for all but the most difficult instances. Indeed, in many cases, the metaheuristics have found good solutions in an early stage of the search process and have spent the remaining time trying to improve this solution. To see this, Figure 2.10 reports the percentage differences between the best solution $(\mathbf{R}^B, \boldsymbol{\kappa}^B)$ (see Algorithms 1 and 2) at various time points relative to the overall best solution (*i.e.*, $(\mathbf{R}^B, \boldsymbol{\kappa}^B)$ after 1000 seconds). We make the following observations from Figure 2.10. First, for all uncertainty sets except the general ellipsoidal and discrete sets, the metaheuristics have found solutions that are within 1% of the overall best after 10 seconds, and there is practically no improvement in the best solution after five minutes. For the general ellipsoid and discrete sets, the solutions are within 2% of the overall

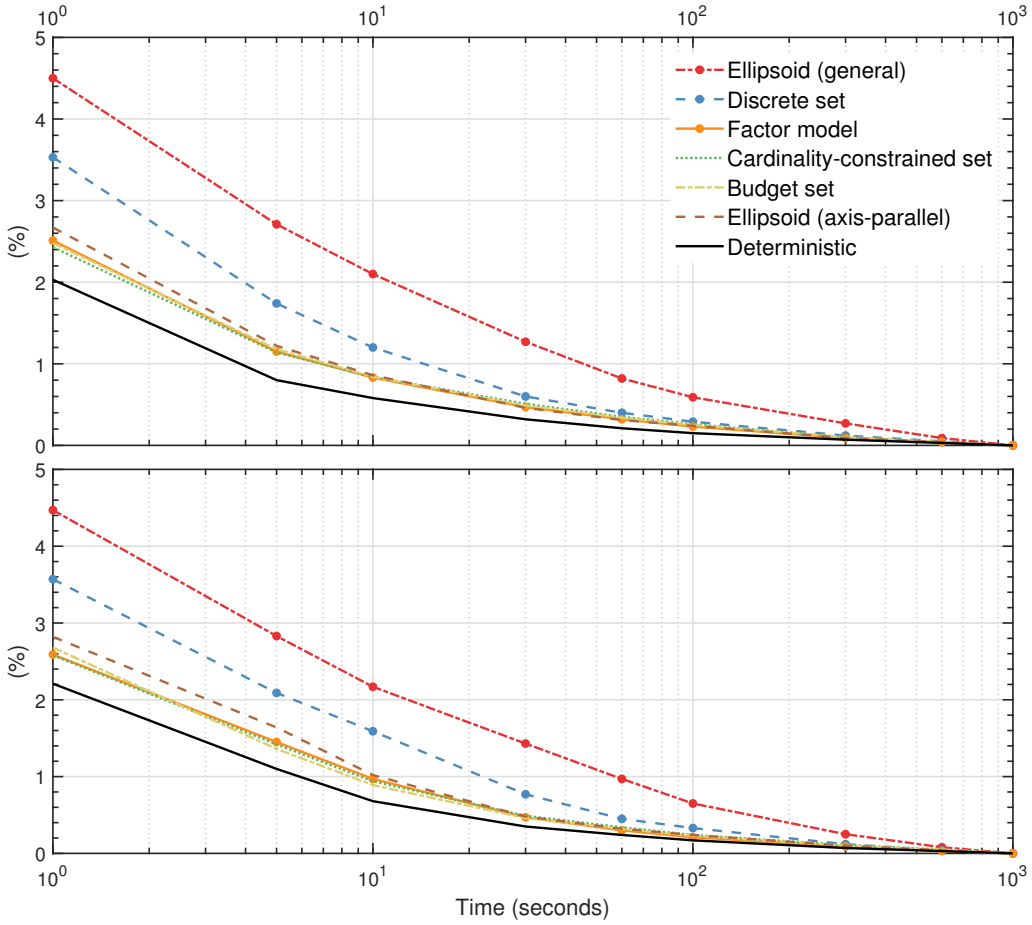
Figure 2.9: Time per local search iteration under different classes of uncertainty sets (normalized with respect to the deterministic problem). The top and bottom graphs show results for the Iterated Local Search and Adaptive Memory Programming algorithms respectively.



best after 10 seconds. This indicates that it is probably better to restart the algorithm with a different random seed, rather than improve the solution, at this point. Second, and similar to Figure 2.9, the performance across the various uncertainty sets correlates well with the complexities reported in Table 2.3, while the performance across the two metaheuristics is similar.

Finally, to understand the sensitivity of the proposed metaheuristic algorithms to the initial random seed, Figure 2.11 plots the average percentage differences of each run relative to the overall best solution of the 10 runs, across all the 74 test instances. The figure shows that both the ILS and AMP algorithms are fairly robust across all classes of uncertainty sets. Indeed, the median deviation is less than 0.2% across all uncertainty sets and in several cases, it is less than 0.1%.

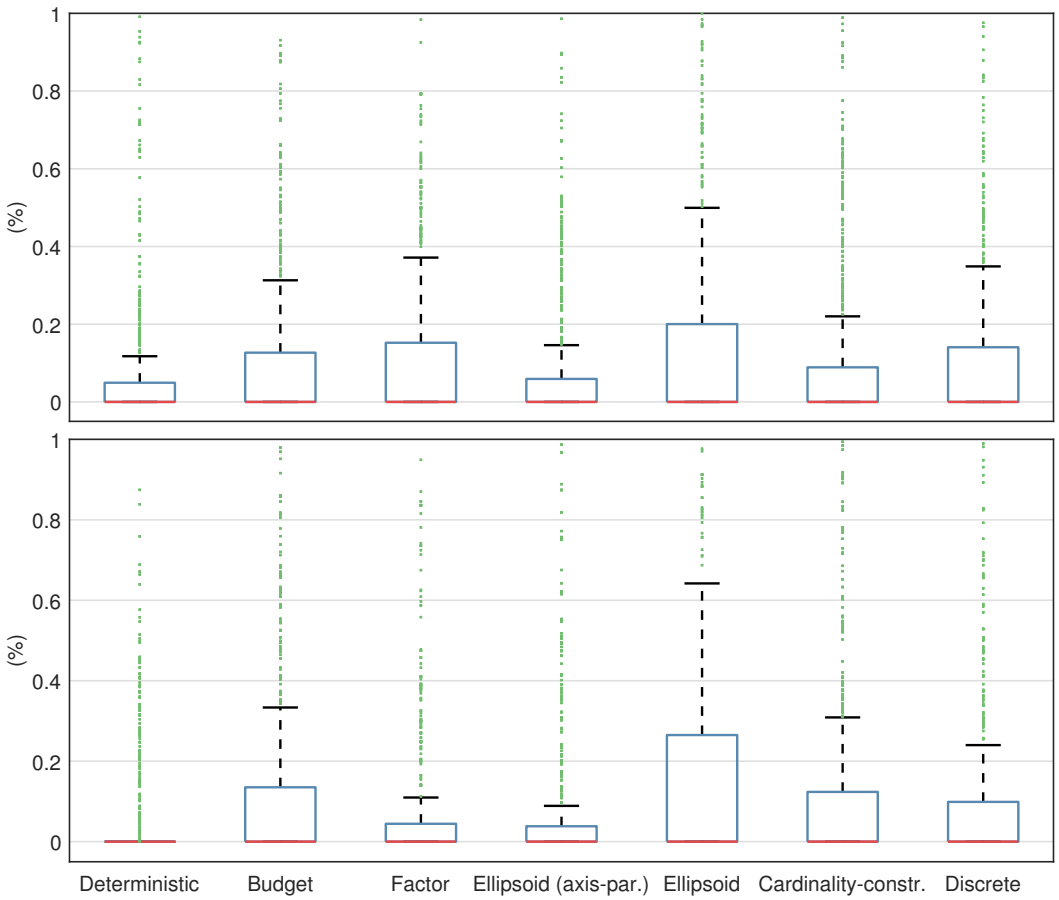
Figure 2.10: Average progress of the metaheuristic solutions under different classes of the uncertainty sets. The top and bottom graphs show results for the Iterated Local Search and Adaptive Memory Programming algorithms respectively. The graphs report the average percentage differences relative to the overall best solution after 1000 seconds.



2.6.3 *Quality of Lower Bounds*

Table 2.4 reports the quality of the lower bounds obtained using the IP formulation (2.13) described in Section 2.5. Specifically, the entries report the guaranteed optimality gaps, which is defined for a given instance as $(z_{ub} - z_{lb})/z_{ub} \times 100\%$, where z_{ub} is objective value of the best solution found across all 10×2 runs of the ILS and AMP metaheuristics after 1000 seconds, while z_{lb} is the global lower bound of the branch-and-cut algorithm after 10,000 seconds. The table shows that the lower bounds for a given uncertainty set $\mathcal{Q}$ are very close to what one can expect for the deterministic problem (indicated by $\{q^0\}$). Indeed, the average optimality gap for the deterministic problem is 5.9%, while the average optimality gap for the robust problem varies between 6.7% and 8.9%, on average. Moreover, the lower bounding is particularly effective for the FSMFD, FSMF and MDVRP variants, *i.e.*, whenever the fleet size is unlimited and either fixed costs or

Figure 2.11: Average deviation of the metaheuristic solutions (with respect to the best solution of 10 runs) under different classes of uncertainty sets. The top and bottom graphs show results for the Iterated Local Search and Adaptive Memory Programming algorithms respectively. The red mark corresponds to the median, the upper edge of the box to the 75^{th} percentile, while the uppermost mark to the maximum of (74×10) runs not considered outliers (which are indicated by green dots).



homogeneous fleets are considered. In such cases, the average optimality gap is less than 5% across all uncertainty sets.

We note that the reported entries in Table 2.4 are very conservative and they are limited by the lower bounds from the branch-and-cut algorithm rather than the upper bounds from the metaheuristics. In fact, we believe that the metaheuristic solutions are near-optimal. There are two reasons for this. First, the branch-and-cut algorithm was never able to find a solution that was better than the provided metaheuristic solutions. Second, we also used our metaheuristics to obtain solutions for all of the 74 original, deterministic benchmarks by setting the vehicle capacities Q_k to their original values. Table 2.5 reports the aggregated results. For each problem variant, the column # reports the number

Table 2.4: Guaranteed optimality gaps of the metaheuristic solutions (in percent) across all benchmarks of the HVRP variants from Table 2.1 and across different classes of uncertainty sets.

	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
HVRPFD	7.25	9.91	9.03	9.03	8.10	11.89	9.10
HVRPD	11.56	13.42	13.23	13.12	12.59	14.73	13.12
FSMFD	3.98	6.43	5.22	6.38	4.95	7.49	6.64
FSMD	8.53	9.68	9.46	9.51	9.34	10.48	9.57
FSMF	2.98	5.65	4.45	4.96	3.72	6.68	5.36
SDVRP	5.30	7.06	6.74	5.90	5.84	7.60	6.19
MDVRP	3.58	5.24	4.86	4.71	4.61	5.80	4.85
All	5.91	7.93	7.28	7.38	6.74	8.92	7.58

of instances, **Gap (%)** reports the average percentage difference between the obtained solution and the *best known solution* (BKS) taken from [214], **BKS found** reports the number of instances for which the obtained solution matched the best known solution and **Time (sec)** reports the average time to find the obtained solution. The table shows that even under the deterministic setting, both metaheuristics are very competitive when compared to existing algorithms for the HVRP (e.g., see [211]). The AMP algorithm is superior to the ILS as it is able to match 64 out of 74 best known solutions with an average gap of 0.1%.

Table 2.5: Summary of results obtained using the metaheuristic algorithms on the 74 original benchmark instances of the deterministic HVRP.

		ILS				AMP			
		Best run		Average		Best run		Average	
	#	Gap (%)	BKS found	Gap (%)	Time (sec)	Gap (%)	BKS found	Gap (%)	Time (sec)
HVRPFD	8	0.19	5	0.31	325	0.19	6	0.30	283
HVRPD	8	0.21	6	0.38	271	0.00	8	0.20	184
FSMFD	12	0.10	8	0.18	244	0.02	9	0.12	192
FSMD	12	0.10	10	0.19	113	0.00	11	0.03	134
FSMF	12	0.09	9	0.20	221	0.02	10	0.10	242
SDVRP	13	0.07	9	0.22	274	0.00	12	0.17	189
MDVRP	9	0.10	4	0.23	228	0.01	8	0.17	183
All	74	0.12	51	0.23	234	0.03	64	0.14	198

2.6.4 *Price of Robustness for Different Classes of Uncertainty Sets*

In this section, we quantify the average increase in total cost of the robust HVRP solution compared to its deterministic counterpart. For each of the five classes of uncertainty sets, we fix $\alpha = 0.1$ and vary β between 0 and 1. For each case, we estimate the cost of the robust solution for each of the 74 test instances using a single run of the AMP metaheuristic under a time limit of 1000 seconds. We then compare the total cost of the obtained solution with that of the deterministic instance (obtained by setting $(\alpha, \beta) = (0, 0)$). Figure 2.12 reports the results of this sensitivity analysis.

Figure 2.12: Average percentage increase in transportation costs relative to the deterministic problem, over different classes of uncertainty sets with $\alpha = 0.1$. The marked square represents the cost increase when the uncertainty set is the n-dimensional hyperrectangle $[(1-\alpha)q^0, (1+\alpha)q^0]$ while the marked ellipse represents the cost increase when the uncertainty set is the axis-parallel ellipsoid that inscribes this n-dimensional hyperrectangle.

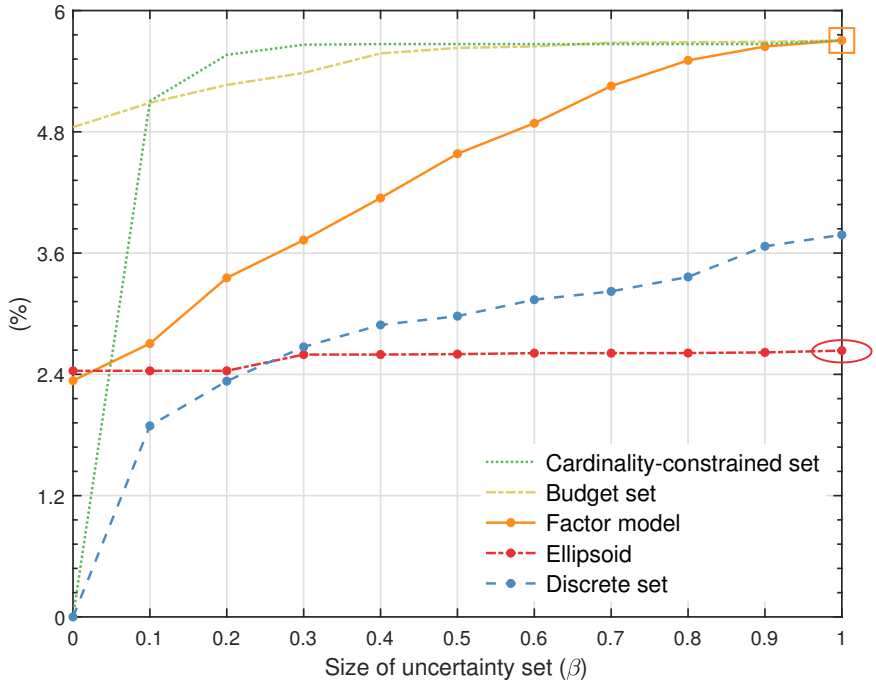


Figure 2.12 offers several interesting insights. First, the robust solutions are only slightly more expensive than their deterministic counterparts. Even when the uncertainty set is a hyperrectangle where every customer can attain their worst realization independently, one can obtain modestly expensive solutions ($\approx 6\%$) at the benefit of being immunized against a considerable random increase in customer demands (up to 10%). This cost increase can be reduced to $\approx 2\%$ by controlling the size and shape of the uncertainty set. Second, taking into account the results in Figure 2.9, it appears that different uncertainty sets offer different levels of tradeoff between cost, robustness and tractability. Indeed,

while discrete sets offer the greatest flexibility in costs, they come at a considerable increase in computational complexity. In contrast, the factor model also appears to offer significant flexibility but at a far less increase in complexity. Finally, the added computational burden in modeling a non-axis-parallel ellipsoid does not pay off in terms of flexibility in transportation costs; indeed, the axis-parallel ellipsoids are only marginally more expensive at the benefit of being extremely tractable in the context of local search.

2.7 SUMMARY

In this chapter, we have attempted to address the challenge of dealing with operational uncertainty during fleet dimensioning and route optimization. In particular, we have studied a broad class of heterogeneous fleet vehicle routing problems where the customer demands are not known precisely when the fleet composition and routes must be decided. These problems are extremely practical since most industrial operations involve heterogeneous fleets. We modeled the unknown customer demands as random variables that can take values in any of five broad classes of practically-relevant uncertainty sets. To hedge against this uncertainty, we aimed to determine a solution that is robust, *i.e.*, remains feasible for all anticipated demand realizations. We elucidated that efficiently computing robust solutions (both in a heuristic and exact manner) lies in the ability to efficiently compute the worst-case loads of vehicle routes over the given uncertainty set. With this insight, we capitalized on well-known local search algorithms in deterministic vehicle routing and augmented them with appropriate data structures to generate robust solutions. We illustrated that the proposed local search can be incorporated in any metaheuristic implementation. Further, the quality of the metaheuristic solutions can be quantified using lower bounds obtained from an augmented integer programming formulation.

Our study offers several novel business insights. First, robust solutions can be obtained with similar computational effort as deterministic solutions. While this is well known in the context of mathematical programming, we have demonstrated that this is also true in the context of metaheuristics. In fact, robust solutions can be obtained by augmenting local search in a modular fashion, and thus it can be readily deployed in commercial codes. Second, the tradeoff between robustness and cost is intimately related to the chosen uncertainty set. While some uncertainty sets might offer greater modeling flexibility or the ability to make better use of available data, they might not necessarily allow a smooth variation in transportation costs as a function of their size. Moreover, the computational tractability of incorporating them in the solution algorithm (whether exact or heuristic) must also be carefully assessed before making a choice.

2.8 APPENDIX: NOMENCLATURE

n	Number of customers
m	Number of vehicle types
$G = (V, E)$	Undirected graph with node set V and edge set E
V_C	Set of customer nodes
K	Set of vehicle types
m_k	Number of available vehicles of type $k \in K$
Q_k	Capacity of vehicle type $k \in K$
c_{ijk}	Routing cost of vehicle type $k \in K$ along edge $(i, j) \in E$
$R = (r_1, \ldots, r_{\lvert R \rvert})$	Vehicle route where $r_l \in V_C$ for all $l \in \{1, \ldots, \lvert R \rvert\}$
$c(R, k)$	Cost of executing route R using vehicle type $k \in K$
$\mathbf{R} = (R_1, \ldots, R_H)$	Set of H vehicle routes $R_1, \ldots, R_H$
$\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_h)$	Fleet composition vector, where $\kappa_1, \ldots, \kappa_H \in K$
$c(\mathbf{R}, \boldsymbol{\kappa})$	Cost of the HVRP solution $(\mathbf{R}, \boldsymbol{\kappa})$
$\bar{c}(\mathbf{R}, \boldsymbol{\kappa})$	Total penalized cost of the HVRP solution $(\mathbf{R}, \boldsymbol{\kappa})$
q_i	Uncertain demand of customer $i \in V_C$
$\mathcal{Q}$	Uncertainty set of customer demands
$\underline{q}, \bar{q}, L, B, b$	Parameters used to define budget sets $\mathcal{Q}_B$
q^0, F, Ψ	Parameters used to define factor models $\mathcal{Q}_F$
q^0, Σ	Parameters used to define ellipsoidal sets $\mathcal{Q}_E$
$q^0, \hat{q}, \Gamma$	Parameters used to define cardinality-constrained sets $\mathcal{Q}_G$
$D, q^{(1)}, \ldots, q^{(D)}$	Parameters used to define discrete sets $\mathcal{Q}_D$
α, β	Parameters used to define the uncertainty sets in Section 2.6
$\chi, \mu, \eta, \nu, \zeta, \delta, \theta$	Parameters used in Algorithms 1 and 2
$(\mathbf{R}^B, \boldsymbol{\kappa}^B)$	Best/incumbent solution in Algorithms 1 and 2
$\mathcal{P}$	Reference set of solutions in Algorithm 2
V_k	Set of customers $i \in V_C$ for which $\max_{q \in \mathcal{Q}} q_i \leq Q_k$, where $k \in K$
$\delta_k(S)$	Subset of edges with one end-point in $S \subseteq V_k$ and other in $V_k \setminus S$
y_{ik}	Binary variable $\in \{0, 1\}$ indicating if customer $i \in V_C$ is visited by a vehicle of type $k \in K$
x_{ijk}	Integer variable $\in \{0, 1, 2\}$ recording the number of times edge $(i, j) \in E$ is traversed by a vehicle of type $k \in K$
e	Vector of ones
$\mathbb{I}[\mathcal{E}]$	Indicator function taking a value of 1 if the expression $\mathcal{E}$ is true and 0 otherwise

2.9 APPENDIX: DETAILED TABLES OF RESULTS

The following tables report the best solutions found for each of the benchmark instances and uncertainty sets that we have considered in this chapter. The budget sets, factor models and general ellipsoids correspond to the setting of $(\alpha, \beta) = (0.1, 0.5)$, the cardinality-constrained and discrete sets correspond to $(\alpha, \beta) = (0.1, 0.2)$, while the axis-parallel ellipsoid corresponds to $(\alpha, \beta) = (0.1, 1.0)$. In each table, "Inst" denotes the instance numbered as per the original dataset, n and m denote the number of customers and vehicle types respectively, while the quantity reported under uncertainty set $\mathcal{Q}$ is the best found solution for that instance and setting of the uncertainty set.

Table 2.6: Summary of results for the HVRPFD instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
13	50	6	2,929.54	3,183.32	3,136.73	3,061.73	3,093.55	3,185.09	3,047.35
14	50	3	9,584.67	10,106.67	10,103.02	9,600.38	9,605.91	10,106.67	9,599.48
15	50	3	2,761.41	3,065.29	2,965.21	2,941.70	2,941.70	3,065.29	2,934.85
16	50	3	3,085.06	3,265.41	3,238.55	3,145.47	3,221.38	3,265.41	3,134.01
17	75	4	1,960.59	2,076.96	2,067.28	2,001.71	2,013.99	2,076.96	1,991.14
18	75	6	3,524.34	3,748.68	3,709.86	3,628.33	3,662.54	3,745.10	3,618.75
19	100	3	9,693.23	10,420.34	10,420.34	9,701.73	9,748.98	10,420.34	9,701.64
20	100	3	4,469.86	4,795.14	4,731.65	4,599.16	4,714.54	4,834.17	4,612.88

Table 2.7: Summary of results for the HVRPD instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
13	50	6	1,432.57	1,517.84	1,502.24	1,481.13	1,482.49	1,517.84	1,468.83
14	50	3	584.38	606.67	603.02	596.53	597.54	606.67	596.63
15	50	3	961.41	1,015.29	1,012.64	991.70	991.70	1,015.29	984.85
16	50	3	1,085.06	1,144.94	1,133.16	1,120.57	1,121.38	1,144.94	1,110.41
17	75	4	1,009.48	1,061.96	1,052.28	1,021.50	1,023.99	1,061.96	1,020.28
18	75	6	1,761.88	1,823.58	1,812.32	1,783.93	1,788.38	1,823.58	1,777.02
19	100	3	1,093.23	1,120.34	1,120.34	1,101.64	1,116.47	1,120.34	1,101.64
20	100	3	1,472.97	1,534.17	1,533.62	1,501.53	1,514.39	1,534.17	1,508.31

Table 2.8: Summary of results for the FSMFD instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
3	20	5	986.93	1,144.22	1,118.76	1,106.84	1,106.65	1,144.22	1,092.59
4	20	3	6,377.28	6,437.33	6,432.25	6,413.06	6,413.06	6,437.33	6,392.34
5	20	5	1,167.40	1,322.26	1,298.72	1,287.48	1,287.48	1,322.26	1,249.08
6	20	3	6,416.11	6,516.47	6,500.82	6,499.74	6,499.74	6,516.47	6,470.82
13	50	6	2,711.61	2,964.65	2,935.40	2,906.68	2,873.24	2,964.65	2,908.96
14	50	3	8,582.05	9,126.90	8,644.00	8,618.16	8,618.16	9,126.90	8,618.16
15	50	3	2,450.82	2,634.96	2,608.61	2,590.47	2,591.86	2,634.96	2,591.93
16	50	3	2,906.71	3,168.92	3,137.50	3,061.09	3,070.34	3,168.92	3,052.39
17	75	4	1,872.49	2,004.48	1,969.64	1,932.82	1,946.09	2,004.48	1,925.47
18	75	6	2,918.45	3,153.09	3,100.08	3,056.88	3,064.50	3,152.16	3,063.70
19	100	3	8,094.97	8,662.86	8,649.99	8,132.14	8,409.14	8,662.86	8,134.19
20	100	3	3,839.11	4,165.91	4,138.25	4,046.05	4,085.22	4,168.44	4,054.18

Table 2.9: Summary of results for the FSMF instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
3	20	5	887.99	954.37	949.79	921.23	927.96	951.61	918.63
4	20	3	6,327.60	6,437.33	6,432.25	6,413.06	6,413.06	6,437.33	6,379.27
5	20	5	919.86	1,005.27	980.57	961.63	963.63	988.63	949.85
6	20	3	6,353.72	6,516.47	6,500.82	6,499.74	6,499.74	6,516.47	6,448.52
13	50	6	2,223.70	2,399.20	2,365.21	2,305.17	2,327.57	2,406.36	2,304.11
14	50	3	8,562.41	9,119.03	8,644.00	8,618.16	8,618.16	9,119.03	8,618.16
15	50	3	2,360.49	2,586.37	2,561.65	2,475.63	2,492.99	2,586.37	2,463.19
16	50	3	2,506.66	2,721.40	2,697.88	2,610.03	2,646.51	2,720.43	2,612.59
17	75	4	1,621.25	1,734.53	1,712.24	1,658.29	1,680.08	1,734.53	1,657.43
18	75	6	2,195.13	2,369.65	2,322.62	2,272.55	2,294.51	2,369.65	2,267.88
19	100	3	8,094.97	8,662.86	8,649.99	8,129.33	8,382.79	8,662.86	8,135.02
20	100	3	3,714.44	4,043.97	3,973.46	3,862.99	3,911.89	4,060.73	3,874.70

Table 2.10: Summary of results for the FSMD instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
3	20	5	555.66	623.22	609.51	604.02	604.02	623.22	585.02
4	20	3	369.77	378.70	373.24	369.77	373.24	380.71	373.24
5	20	5	713.62	742.87	737.36	736.90	736.90	742.85	721.00
6	20	3	391.42	414.66	415.03	396.26	405.46	406.19	396.26
13	50	6	1,405.87	1,491.86	1,488.78	1,468.28	1,471.82	1,491.86	1,460.06
14	50	3	575.63	603.21	601.71	583.94	588.64	603.21	583.94
15	50	3	944.96	999.82	997.98	979.40	985.19	999.82	974.59
16	50	3	1,078.67	1,131.00	1,114.78	1,110.88	1,111.69	1,131.00	1,107.74
17	75	4	1,006.50	1,036.54	1,029.75	1,017.52	1,020.50	1,038.60	1,016.64
18	75	6	1,746.53	1,800.80	1,795.02	1,778.61	1,785.58	1,800.80	1,775.35
19	100	3	1,073.71	1,106.17	1,101.07	1,084.54	1,099.69	1,105.44	1,093.74
20	100	3	1,468.79	1,530.52	1,525.33	1,495.31	1,514.46	1,533.24	1,497.01

Table 2.11: Summary of results for the SDVRP instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
1	50	3	609.52	640.32	634.19	621.50	624.77	640.32	623.37
2	50	2	563.85	598.10	593.59	569.45	577.70	598.10	569.45
3	75	3	899.90	954.32	938.11	908.95	922.33	954.32	908.95
4	75	2	806.72	854.43	843.73	831.60	836.25	854.43	825.27
5	100	3	963.09	1,003.57	993.64	977.14	978.00	1,003.57	976.01
6	100	2	965.86	1,028.52	1,016.95	987.88	1,000.59	1,028.52	983.46
7	27	3	391.30	391.30	391.30	391.30	391.30	391.30	391.30
8	54	3	664.46	664.46	664.46	664.46	664.46	664.46	664.46
9	81	3	948.23	948.23	948.23	948.23	948.23	948.23	948.23
10	108	3	1,189.69	1,218.75	1,218.75	1,208.74	1,208.74	1,218.75	1,200.34
13	54	3	1,150.31	1,194.18	1,194.18	1,162.85	1,171.58	1,194.18	1,171.58
14	108	3	1,873.74	1,960.88	1,960.88	1,886.53	1,908.76	1,960.88	1,889.59
23	100	3	764.19	803.29	803.29	781.81	784.83	803.29	781.81

Table 2.12: Summary of results for the MDVRP instances.

Inst	n	m	$\{q^0\}$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E^{\text{ax}}$	$\mathcal{Q}_E^{\text{gen}}$	$\mathcal{Q}_G$	$\mathcal{Q}_S$
1	50	4	560.63	576.87	570.81	570.81	570.81	576.87	575.36
2	50	4	464.35	473.53	473.01	464.48	464.48	473.22	464.48
3	75	5	623.79	640.65	635.93	629.86	630.98	641.19	629.67
4	100	2	951.28	999.21	997.20	973.68	982.38	999.21	978.74
5	100	2	725.55	750.03	747.29	731.27	741.44	750.03	732.17
6	100	3	838.00	876.50	871.47	864.49	867.39	877.26	865.93
7	100	4	843.30	881.97	871.31	855.39	860.65	881.97	855.55
12	80	2	1,290.93	1,318.95	1,314.36	1,314.36	1,314.36	1,318.95	1,301.68
15	160	4	2,463.14	2,505.42	2,505.42	2,505.42	2,505.42	2,505.42	2,492.55

3

TACTICAL PLANNING UNDER CUSTOMER ORDER UNCERTAINTY

The previous chapter considered problems in which the vehicles visit a fixed set of customers (with variable order size) that are all known in advance. In this chapter, we study problems in which the customers who will be visited over a tactical planning horizon are not fully known in advance. Specifically, customers can place their requests dynamically in any period, and the aim is to determine a visit schedule and associated routing plan for each period without the precise knowledge of the customer base. Naturally, a key challenge is to generate routing plans that can flexibly accommodate potential requests that have not yet been placed.

To tackle this problem, we model future potential customer requests as binary random variables, and seek to determine a visit schedule that remains feasible for all anticipated realizations of the requests. The planning process can therefore be viewed as a multi-stage robust optimization model, where the (discrete recourse) decision to serve requests on each day of the horizon is a function of all the orders that have been received up to that day. However, since solving this model is generally intractable, we approximate it via a non-anticipative two-stage model for which we propose a novel solution approach. We investigate the quality of the solutions we obtain in two ways: *(i) theoretically*, we derive a valid lower bound on the multi-stage model, and *(ii) empirically*, we perform Monte Carlo simulations on a rolling horizon to show that it significantly outperform existing approaches in terms of both operational costs and fleet utilization.

The remainder of this chapter is structured as follows. Section 3.1 provides background information, in terms of the industrial application that motivated this work as well as related literature. Section 3.2 provides a mathematical definition of the problem that we are contemplating, the uncertainty set, the multi-stage robust optimization model, as well as the two-stage models that provide conservative (*i.e.*, upper bound) and progressive (*i.e.*, lower bound) approximations of the latter. Section 3.4 presents an integer programming formulation of the conservative two-stage model and discusses its solution through a branch-and-cut algorithm, while Section 3.5 presents a scheme for the computation of lower bounds via the progressive approximation. Finally, Section 3.6 presents computational results on benchmark problems, while we summarize the key results from this chapter in Section 3.7.

3.1 BACKGROUND

Traditional variants of the VRP are of an operational nature and the typical setting involves routing within a single period, e.g., a work day. However, several transportation problems arising in practice are of a tactical nature, since they involve the routing of vehicles over multiple service periods, e.g., a week. This is particularly the case when customers may dynamically place service requests on any given day of the week, and each request specifies a set of future days during which service can take place. At the end of each day, the distributor must make scheduling decisions to assign a visit day to each unfulfilled service request, along with the standard routing decisions, so as to minimize the long-term transportation costs. The tactical plan is implemented in a rolling horizon fashion: only routes of the first day are executed while new service requests are received; unfulfilled requests at the end of the first day and new requests accumulated during the day constitute the new portfolio of orders to be considered for scheduling the following day. Such decision-making setups are typical in systems in which services are provided by appointment, and the following section describes one such system.

3.1.1 *Industrial Motivation*

Our research is motivated by the business setting of an industrial gases company, whose main production process involves separating air into its components, primarily nitrogen, oxygen and argon, which are then used for a wide variety of industrial, medical, retail and other purposes. After production, these gases are filled in cylinders that are to be transported to the customers using trucks. Distribution operations involve receiving orders from customers during the day. In addition to the order volume, customers in certain markets specify earliest and latest acceptable visit dates at the time of placing their order. The resulting "day windows" are allowed to be open, so that the distributor does not need to commit to a delivery date at the time of order placement. Therefore, on any given day of operation, the goal is to decide which unfulfilled orders to serve and which ones to leave for future days. The delivery schedules themselves are generated by solving a traditional VRP, considering constraints on the number of trucks, their capacity, as well as driver availability.

During this decision-making process, it is crucial to anticipate future customer orders and explicitly hedge against their underlying uncertainty. Indeed, ignoring the possibility that customers will place orders in the future can lead to infeasible situations, e.g., the number of vehicles required may be more than what is available. This situation could arise because too many orders were postponed until their latest acceptable dates, and huge costs must now be incurred to recover feasible schedules (e.g., additional vehicles must be commissioned or drivers must be paid overtime). The alternative is to serve orders beyond their acceptable dates; however, this is typically avoided, since

it is perceived as poor customer service and has a negative impact on the company's reputation.

The above description of the VRP is representative of the problem and its complexities at other companies as well as industries. Relevant examples include scheduling of maintenance personnel [61, 247, 257], blood delivery to hospitals [9, 149], food distribution and collection [100, 267], courier services [18], auto-carrier transportation [89], as well as distribution operations arising in city logistics [14]. Fortunately, companies that are faced with such operations often have significant amounts of historical data, which can be used to obtain demand forecasts and provide information regarding calls for service in future time periods. The objective of this work is to contribute a decision support methodology that can use this information and generate risk-averse schedules for the tactical planning of multi-period vehicle routing operations.

3.1.2 *Related Work*

A tactical level multi-period routing problem was first introduced by Angelelli, Grazia Speranza, and Savelsbergh [11] and Angelelli, Savelsbergh, and Grazia Speranza [12], who considered the problem in which a number of customer requests are received at the beginning of each day, and each of these requests must be served using a single uncapacitated vehicle either in the day it was received or in the following day. The decision-maker must thus decide at the beginning of each day which unfulfilled requests to serve during that day and which ones to postpone to the future so as to minimize the sum of total routing costs across the planning horizon. The problem was extended in [267] (where it was referred to as the *Dynamic Multi-Period VRP*) and in [32] (where it was referred to as the *Tactical Planning VRP*); in both cases, the authors considered multiple capacitated vehicles and the possibility for customers to request a service day window spanning more than two days. Further extensions to these problems include the consideration of arrival-time windows within each visit day [19] as well as inventory holding costs at customer service locations [14]. We remark that these multi-period routing models are closely related to the *Periodic VRP* [118, 156], in which customers specify allowable visit day combinations and service frequencies over a short-term planning horizon (typically one week) and the decision-maker attempts to meet these service requirements while minimizing routing costs. A key difference is that the Periodic VRP is a strategic decision-making problem because, in practice, the weekly routing plan is operated unchanged over the course of several months and all information (customer demand, in particular) is available at the beginning of the planning horizon.

In all of the aforementioned works, decisions are determined through the solution of deterministic optimization problems by considering only some nominal scenario of future customer orders (e.g., taking into account only those service requests that have already been placed and ignoring the potential for customers to place new or augmented service requests at some future point in time). As we have already discussed, such

decisions can create situations which can either be infeasible, or too expensive in terms of transportation costs. Therefore, in the remainder of this section, we only review those papers that explicitly treat uncertainty in vehicle routing problems.

One option for taking into account the uncertainty in future service requests is stochastic programming, which models the uncertain parameters of an optimization problem as random variables with known probability distributions [58]. Over the last four decades, there has been a rich development of stochastic programming models for several variants of the VRP under uncertainty. Gendreau, Jabali, and Rei [125] provide an excellent overview of the existing models for a variety of uncertain parameters; we mention here only those papers which study the case of uncertain customer orders, which is known in the literature as the *VRP with Stochastic Customers* (VRPSC). The VRPSC is typically formulated as a two-stage model, where the first stage decisions (designed before the realization of customer orders) consist of designing feasible vehicle routes that visit all potential customer requests, while the second stage recourse decisions (selected after the realization of customer orders) consist of following the designed vehicle routes, while skipping those customer requests that did not materialize. This model was first introduced by Jaillet [158] for the case of a single vehicle, and since then, solution approaches have been proposed for that model as well as its extensions by Bertsimas [50], Gendreau, Laporte, and Séguin [126], and Laporte, Louveaux, and Mercure [178]. We remark that the VRPSC is an operational model with a planning horizon of a single time period.

In the context of multi-period VRPs, the study by Albareda-Sambola, Fernández, and Laporte [8] considers probabilistic descriptions of customer order uncertainty. The authors assume that on any given day, the probability of a potential customer requesting service at any point in the future is known precisely. However, rather than model the problem as a stochastic program, the authors utilize this information to formulate an ad-hoc *Prize Collecting VRP* over the known customer orders, which aims to decide at the beginning of each day which requests to serve along with the actual vehicle routes; the prize for each known customer order is heuristically set according to a function that increases with respect to the order's temporal proximity to its service deadline and decreases with respect to its spatial proximity to uncertain future orders.

The tactical planning multi-period VRP that we study in this chapter may also be classified as a VRP of *dynamic* nature, because not all customer requests that will be served over the planning horizon are known in advance, being gradually realized during the execution of the tactical plan. There is a huge body of literature on dynamic VRPs, and we refer the reader to [35] for a survey of these problems. A key difference between the traditional family of dynamic VRPs and tactical planning VRPs is that the former are of an operational nature, and are characterized by a high *degree of dynamism* [183]; that is, the frequency at which new information is obtained and reacted upon is significantly higher (of the order of hours and minutes, as opposed to days), thereby reducing the time available for optimization computations and, hence, the solution strategy (and, often, the solution quality). Moreover, the primary decisions often involve real-time

re-routing of vehicle schedules during their execution (e.g., see [39, 227]) or re-scheduling multiple trips using the same vehicle (e.g., see [22, 169]) rather than serving all pending customer orders. Similarly, the typical objective is to maximize the number of customer orders served and minimize service times, rather than optimize transportation costs. Nevertheless, in the context of multi-period VRPs, such decision-making setups have been studied by [10], who devised purely "online" re-optimization approaches that ignore uncertainty, and more recently by [258], who used approximate dynamic programming techniques that explicitly account for customer order uncertainty. In both cases, the authors consider a variant of the multi-period VRP in which the decision-maker may additionally choose to incorporate an arriving customer order into the vehicle schedule currently in execution or postpone its service to the next day.

In contrast to the above approaches, robust optimization is an alternative framework that could be used for decision-making under uncertainty in this context. Similar to stochastic programming, robust optimization models the uncertain parameters of an optimization problem as random variables, but instead of describing them stochastically via probability distributions, it requires only knowledge of their support. The basic robust optimization problem consists of determining a solution that remains feasible for any realization of the uncertain parameters over this prespecified support, also referred to as the uncertainty set. We refer the reader to [36] and [51] for a detailed review of the robust optimization literature.

Over the last decade, several classical variants of the VRP under uncertainty have been studied through the lens of robust optimization. In particular, these include the classical *Capacitated VRP* under demand uncertainty [109, 137, 138, 207, 244] and the *VRP with Time Windows* under travel-time uncertainty [6]. Apart from these VRP variants, robust optimization has also been used to address some related arc routing problems under service-time uncertainty [78] and inventory routing problems under demand uncertainty [55, 239]. On a related note, Jaillet, Qi, and Sim [159] proposed a new risk measure, called the *requirements violation index*, as a criterion to evaluate how well a candidate VRP solution meets its constraints under a *distributionally robust* model of uncertainty, in which parameters are described via (possibly ambiguous) probability distributions. However, in contrast to robust optimization, which determines a minimum-cost solution subject to a budget constraint on the uncertainty, their method determines a minimum-risk solution subject to a budget constraint on the cost. Moreover, their approach requires the uncertain attribute (e.g., vehicle load) to be an affine function of the underlying uncertainties (e.g., customer demand). Recently, Zhang et al. [278] generalized this requirement to piecewise affine functions, and addressed travel-time uncertainty. To the best of our knowledge, none of the aforementioned approaches addresses uncertainty in customer orders. This is particularly challenging in the context of robust optimization because uncertain parameters such as demand and travel times are typically modeled as continuous (as opposed to discrete) random variables, allowing the reformulation of the corresponding robust optimization models to a finite-dimensional deterministic model, which can be solved relatively efficiently. In contrast, the presence

or absence of a customer order is a discrete event, providing more challenges for robust optimization modeling and solution approaches.

In recent years, robust optimization has also been extended to solve multi-stage decision-making problems, in which a sequence of uncertain parameters is observed over time and the decision-maker can take recourse actions whenever the value of an uncertain parameter becomes known. Besides faithfully modeling the dynamic nature of decision-making processes in practice, multi-stage problems are essential to mitigate the conservatism of traditional single-stage (also known as *static*) robust optimization problems. However, while multi-stage problems involving continuous recourse decisions have been well studied [38, 79], the literature on robust optimization with discrete recourse decisions is relatively sparse. Zhao and Zeng [280] have developed a generalized *column-and-constraint generation* framework to address two-stage problems in a fully adaptive fashion, in which the resulting first-stage solution is no more conservative than any other robust feasible solution. Other approaches are concerned with the design of conservative approximations of the true multi-stage problem and fall into one of three categories: (i) *decision rule* approaches, which model the recourse decisions as explicit functions of the uncertain parameters [45, 54], (ii) *K-adaptability* approaches in which the decision-maker designs K sets of discrete recourse decisions in the first-stage and implements the best design after observing the realization of uncertain parameters [52, 145], and (iii) *uncertainty set partitioning* approaches, which simulate the recourse nature of discrete decisions by designing separate sets of decisions for different, pre-specified subsets of the uncertainty set [53, 218].

3.1.3 *Our Contributions*

In this chapter, we study the modeling and solution of the multi-period VRP under customer order uncertainty, casting it as a multi-stage robust optimization model. To that end, we model uncertain customer orders as binary random variables that have realizations in an uncertainty set of finite (but possibly very large) cardinality, where each member of the set corresponds to a combination of customer orders that might potentially realize over the planning horizon. This set constitutes a flexible representation, allowing us to capture practically meaningful scenarios that adhere to underlying correlations linking customer requests, and can be easily regressed from historical data without requiring detailed probability distributions.

However, since the numerical solution of a multi-stage model can prove challenging in practice, we propose to approximate it with a tractable, non-anticipative two-stage counterpart. We also devise a partially-anticipative two-stage model that provides lower bounds on the optimal value of the multi-stage model, and establish conditions under which the solutions provided by the two models coincide. Finally, we conduct a thorough computational study to elucidate the numerical tractability of our algorithm, the approxi-

mation quality of the non-anticipative two-stage model, and the closed loop performance of the solutions provided by this model in a rolling horizon simulation.

The solution of the two-stage model is aided by algorithmic efficiencies that we propose to improve the generalized column-and-constraint generation framework [280] for two-stage robust optimization problems with binary recourse decisions. These improvements are particularly suited in cases when there are no second-stage costs; that is, when the recourse problem is a mere feasibility problem.

3.2 PROBLEM DEFINITION

Let Π denote a (possibly infinite) time horizon, whose elements represent time periods (days).[1] On any given day $d \in \Pi$, a number of customers place a service request. The set of all customers who can request service during Π is assumed to be known and denoted by N. Each $i \in N$ is associated with quantities $q_i \in \mathbb{R}_+$, $e_i \in \mathbb{Z}_+$ and $\ell_i \in \mathbb{Z}_+$, where $1 \leq e_i \leq \ell_i$, which have the following meaning: if i places a service request on day d, then a demand *quantity* q_i must be delivered to i no *earlier* than e_i days after d, and no *later* than ℓ_i days after d; that is, service must be provided in the day window $\{d + e_i, \ldots, d + \ell_i\}$.[2][3] For notational convenience, we shall define the *width* of this day window to be $w_i := \ell_i - e_i + 1$. Any such pair $v = (i, d)$ represents a *customer order* that is associated with demand quantity $q_v \equiv q_i$ and service day window $P_v \equiv \{d + e_i, \ldots, d + \ell_i\}$.[4]

Let $G = (N', E)$ denote an undirected graph with nodes $N' = N \cup \{0\}$ and edges E. Node 0 represents the unique depot, which is equipped with m homogeneous vehicles, each of capacity $Q \in \mathbb{R}_+$ and available on every day of the horizon. We denote the set of vehicles as $K = \{1, \ldots, m\}$. Each vehicle incurs a traveling cost $c_{ij} \in \mathbb{R}_+$, if it traverses the edge $(i, j) \in E$. We define $c_{ii} = 0$ for all $i \in N$. For ease of notation, given two customer orders $u = (i, d)$ and $v = (j, d')$, we let c_{uv} mean the same thing as c_{ij}.

Let $0 \in \Pi$ denote the (end of the) current time period, and $V_0 \subseteq \{(i, p) \in N \times \Pi : p \leq 0\}$ the set of *pending orders*, i.e., orders that were received in the past but have not yet been served. Similarly, for any $p \geq 1$, let $V_p = N \times \{p\}$ denote the set of *potential future orders* that may be received in period $p \in \Pi$. Our goal is to determine a feasible visit schedule

1 Throughout the chapter, the terms *days* and *periods* are used interchangeably.

2 The requirement $1 \leq e_i$ is typical in many operations, where orders cannot be served on the same day in which they were placed. This is often due to the fact that available vehicles have already been loaded earlier in the day, and have departed the depot to serve other customers before the time the order is placed.

3 We remark that our proposed method allows the service period to be defined as any subset of Π, and not necessarily consecutive days constituting a window. However, for ease of exposition, we do not present this generalization.

4 Observe that, as per this definition, if a customer i places a service request more than once during the horizon, on days d and d', then the orders (i, p) and (i, p') are treated independently. Thus, although it is possible to associate different demand quantities q_{id} and $q_{id'}$ to these orders, we do not consider this possibility for ease of exposition.

$(S_1, S_2, \dots,)$ over the future horizon $\{1, 2, \dots,\}$ that services all pending orders in V_0 as well as future orders from $\{V_p\}_{p\geq 1}$, in a way that minimizes long-term costs; here, S_p denotes the set of orders selected to be served on day p. In view of this goal, we shall restrict our attention to a finite planning horizon $P = \{1, \dots, h\}$ consisting of the $h \geq 1$ subsequent days, and attempt to determine a visit schedule $(S_1, \dots, S_h)$ over P. The cost of this schedule is determined by computing a vehicle routing plan that services the orders in S_p, for each $p \in P$. Before we formally describe our model, we remark that, in practice, the computed schedule $(S_1, \dots, S_h)$ will be implemented in a *rolling horizon* fashion: only the vehicle routes corresponding to S_1 will be executed; new orders received on day 1 will be recorded, V_0 will be updated, and the entire procedure will be repeated over the updated horizon $\{2, \dots, h+1\}$. Therefore, in the following sections, we shall only focus on the modeling and solution procedure of the planning problem over P. We shall return to the rolling-horizon context in Section 3.6, where we evaluate the performance of our proposed method using rolling horizon simulations.

Because of the finiteness of the planning horizon P, we can make some simplifying assumptions without loss of generality. First, we shall assume that the day window P_v, of any pending order $v = (i, d) \in V_0$, is updated so that it satisfies $P_v = \{\max\{1, d + e_i\}, \dots, d + \ell_i\}$.[5] Second, we shall assume that the set of all (pending and potential) orders $V := V_0 \cup V_1 \cup \dots \cup V_h$ is preprocessed such that any order $v \in V$ satisfies $d + \ell_i \leq h$;[6] along with the requirement that customers cannot be served on the day they requested service, i.e., $e_i \geq 1$, this means that $V_h = \emptyset$. These assumptions imply that, after preprocessing, we have $P_v \subseteq P$ for all orders $v \in V$.

3.2.1 *Uncertainty Model*

In practice, it is unlikely that all potential future orders from $\{V_p\}_{p\in P}$ will *materialize* (i.e., be received) during the planning horizon. Therefore, these orders are *uncertain* in the context of the current planning problem. To capture this uncertainty, we model the presence (or absence) of future orders as binary random variables ξ, and assume only that their support $\Xi \subseteq \{0,1\}^{|V|}$ is known. Specifically, ξ_v (equivalently referred to as ξ_{id}) is an uncertain parameter attaining the value of one, if the order $v = (i, d) \in V$ materializes (i.e., customer i places an order on day d), and zero otherwise. Note that $\xi_v = 1$ for all $v \in V_0$, and hence, these components of ξ are deterministic. For notational convenience, we define $\xi^0 := (\xi_v)_{v\in V_0}$ and $\xi^p := (\xi_v)_{v\in V_p}$ to be the restriction of the vector ξ to those orders that are known to be pending at the beginning of the planning horizon and to those orders that can potentially materialize in period $p \in P$, respectively. We also define $\xi^{[p]} := (\xi^0, \dots, \xi^p)$ as the parameter restriction up to period p; and, $\Xi^{[p]} := \left\{\xi^{[p]} \in \{0,1\}^{|V_0|+\dots+|V_p|} : \xi \in \Xi\right\}$ as the corresponding projection of Ξ,

5 $v \in V_0$ is an unfulfilled order. Therefore, if $d + e_i < 1$, then its day window has to be shrunk to $\{1, \dots, d + \ell_i\}$.
6 Orders v for which $d + \ell_i > h$ will be served outside the horizon and can be safely removed from consideration.

for all $p \in \{0, 1, \ldots, h\}$. Finally, we denote by $\hat{\xi}$ the *nominal realization* of the uncertain parameters, which corresponds to the scenario where only the pending orders need to be served and no other customer orders are received during the planning horizon; that is, $\hat{\xi}_v = 1$, if $v \in V_0$, and 0 otherwise. Throughout the chapter, we shall assume that the support Ξ, also referred to as the *uncertainty set*, satisfies the following conditions:

(C1) The uncertainty set Ξ is non-empty. In particular, $\hat{\xi} \in \Xi$.

(C2) The pending customer orders are a part of every uncertainty realization. Stated differently, $\Xi^{[0]} = \{\mathbf{1}\}$, where $\mathbf{1} \in \mathbb{R}^{|V_0|}$ denotes the vector of ones.

(C3) For each order $v \in V$, we have $\max\{\xi_v : \xi \in \Xi\} = 1$.[7]

We remark that, due to the finiteness of $\{0,1\}^{|V|}$, every uncertainty set which satisfies the above conditions admits a polyhedral description of the form:

$$\Xi = \left\{ \xi \in \{0,1\}^{|V|} : \xi^0 = \mathbf{1}, \ \sum_{p \in P} A_p \xi^p \leq b \right\}, \text{ where } A_p \in \mathbb{R}^{r \times |V_p|} \text{ and } b \in \mathbb{R}^r_+. \quad (3.1)$$

CONSTRUCTING THE UNCERTAINTY SET FROM DATA. We provide some guidance on how an uncertainty set can be constructed in practice, including when historical data might be available. We focus our attention to the class of *budgeted uncertainty sets* which have the following form:

$$\Xi_B = \left\{ \xi \in \{0,1\}^{|V|} : \xi^0 = \mathbf{1}, \ \sum_{v \in B_l} \xi_v \leq b_l \text{ for } l \in \{1, \ldots, L\} \right\} \quad (3.2)$$

Here, $L \in \mathbb{N}$, $B_l \subseteq V \setminus V_0$ and $b_l \in \mathbb{N}$ are parameters that need to be specified. The l^{th} inequality imposes a limit b_l on the total number of customer orders that can be received from the set B_l, and thus represents a *budget of uncertainty*. Observe that, by setting $b_l = 0$, for all $l \in \{1, \ldots, L\}$, the uncertainty set $\Xi_B = \{\hat{\xi}\}$ reduces to a singleton, corresponding to the nominal realization. As the values of b_l increase, the size of the uncertainty set $|\Xi_B|$ enlarges and more scenarios of future service requests are considered. When $b_l = |B_l|$ for all $l \in \{1, \ldots, L\}$, Ξ_B becomes a hypercube and all potential future orders are considered. We shall refer to Ξ_B as a *disjoint budget uncertainty set*, if the sets $\{B_l\}_{l=1}^L$ are disjoint; that is, $B_l \cap B_{l'} = \emptyset$ for $l \neq l'$. The disjoint budget structure will play an important role later, both in gaining a better theoretical understanding as well as enabling efficient algorithms. Practical examples of (not necessarily disjoint) budgets that are motivated in the context of the multi-period VRP are presented in the following.

(a) *Budget of orders received during the planning horizon.* This is obtained by setting $L = 1$ and $B_1 = V \setminus V_0$. Here, b_1 represents the maximum total number of orders received during the planning horizon. Observe that this is precisely the *cardinality-constrained uncertainty set* proposed by Bertsimas and Sim [57], and thus the latter constitutes a special case of (3.2).

7 This condition can always be ensured by removing redundant orders (e.g., from customers who may never place a service request on a particular day) from consideration.

(b) *Budgets of orders received on individual days.* This is obtained by setting $L = h$ and $B_p = V_p$ for all $p \in P$. Here, b_p represents the maximum number of orders received on day p.

(c) *Budgets of orders received from individual customers.* This is obtained by setting $L = |N|$ and $B_i = \{(i, p) \in V : p \in P\}$ for all $i \in N$. Here, b_i represents the maximum number of orders placed by customer i during the planning horizon.

The budget parameter b in each of the above cases can be computed either from domain knowledge or using statistical models. As an example of the latter, consider the following two cases.

– *Independent orders (as in case (b)).* Suppose that ξ_v, $v \in B_l$, are independent binary random variables with probabilities $\alpha_v \in [0, 1]$, that have been estimated from data. Then, the sum $\sum_{v \in B_l} \xi_v$ follows a *Poisson binomial distribution* with parameters $(\alpha_v)_{v \in B_l}$. If F denotes its cumulative distribution function, then the inequality

$$\sum_{v \in B_l} \xi_v \leq F^{-1}(\gamma) \tag{3.3}$$

is satisfied with probability γ. The above inequality can be incorporated as a budget by setting $b_l = F^{-1}(\gamma)$. If $|B_l|$ is large enough, then one can also employ a limit law, such as the central limit theorem, to argue that the inequality

$$\frac{1}{\sigma_l} \sum_{v \in B_l} (\xi_v - \alpha_v) \leq \Phi^{-1}(\gamma), \text{ where } \sigma_l^2 = \sum_{v \in B_l} \alpha_v (1 - \alpha_v) \tag{3.4}$$

is satisfied with probability γ. Here, Φ is the cumulative distribution function of the standard normal random variable (see also [34]). It is easy to see that this inequality can also be incorporated as a budget by appropriately defining b_l.

– *Dependent orders (as in case (a) or case (c)).* If ξ_v, $v \in B_l$, are dependent binary random variables, then one can use tail bounds of $\sum_{v \in B_l} \xi_v$, obtained via simulations, to determine values for the budget parameter b_l. For example, in case (c), in which B_i is a set of temporally distributed orders from customer i, one can simulate the following k^{th} order autoregressive logistic model to determine b_i. Here, $a_{i0}, \ldots, a_{ik}$ are parameters to be estimated from data.

$$\alpha_{ip} = \frac{1}{1 + \exp(a_{i0} + \sum_{j=1}^{k} a_{ij} \xi_{i,p-j})} \tag{3.5}$$

Remark 3.1. *In all of the aforementioned cases, the resulting budget sets Ξ_B can be updated in a rolling horizon context. For example, if customer i has just placed a service request, then the budget $\sum_{v \in B_i} \xi_v \leq 0$ can be imposed to reflect the expectation that i is unlikely to place an order in the next planning horizon. More generally, the probabilities α in (3.3), (3.4) and logistic coefficients a in (3.5) can be estimated in a Bayesian fashion, to obtain improved estimates of the budget parameters b.*

The aforementioned uncertainty sets exploit the binary-valued nature of the uncertain parameters ξ. We remark that there exists a large body of work on uncertainty set construction for general robust optimization problems with continuous-valued uncertain parameters. We refer interested readers to [36, 56, 135] for theory, applications and practical recommendations regarding this subject.

3.2.2 *Multi-Stage Adaptive Robust Optimization Model*

We first describe the deterministic version of the problem. In this regard, let $\xi \in \Xi$ denote a given realization of customer orders over the planning horizon. Given any subset of orders $S \subseteq V$, let $\texttt{CVRP}(S, \xi, t)$ denote the optimal objective value of an instance of the *Capacitated Vehicle Routing Problem* with nodes $\{i \in S : \xi_i = 1\} \cup \{0\}$, travel costs c, demands q, and using at most t vehicles of capacity Q located at the depot node 0. Similarly, let $\texttt{BPP}(S, \xi)$ denote the optimal objective value of an instance of the *Bin Packing Problem*, where the bin size is Q and the items are the elements of $\{i \in S : \xi_i = 1\}$ with corresponding weights q_i.[8] Finally, let $\Delta_p := \{v \in V : p \in P_v\}$ denote the set of all (pending and potential) customer orders that can be serviced in period $p \in P$ of the planning horizon; and, let $\mathcal{F} := \{(S_1, \ldots, S_h) : S_p \subseteq \Delta_p \; \forall \, p \in P, \; S_p \cap S_{p'} = \emptyset \; \forall \, p, p' \in P : p \neq p', \; \cup_{p \in P} S_p = V\}$ denote the set of feasible assignments of customer orders to periods; that is, for each partition $\mathbf{S} = (S_1, \ldots, S_h)$ of V such that $\mathbf{S} \in \mathcal{F}$, S_p is the (possibly empty) subset of orders assigned to period p. For a given vector $\xi \in \Xi$, the *deterministic* problem is:

$$\begin{aligned} \underset{\mathbf{S}}{\text{minimize}} \quad & \sum_{p \in P} \texttt{CVRP}(S_p, \xi, m) \\ \text{subject to} \quad & (S_1, \ldots, S_h) \in \mathcal{F}. \end{aligned} \tag{$\mathcal{DET}(\xi)$}$$

Existing solution methods for the multi-period VRP (e.g., see [32, 267]) attempt to solve the deterministic problem $\mathcal{DET}(\hat{\xi})$ corresponding to the nominal realization $\hat{\xi}$, and effectively assume that no orders other than the currently pending ones will be serviced during the planning horizon. The consequence of this assumption is that the determined solution $\mathbf{S}$ can become infeasible under customer order realizations other than the nominal (refer to Section 3.2.5 for a discussion and to Section 3.3 for an illustrative example). Therefore, in the following, we present a robust optimization model that explicitly hedges against customer order uncertainty.

In this adaptive, multi-stage robust optimization model, the goal is to select the set of customer orders to be served in period 1 in a *here-and-now* fashion, whereas the set of customer orders to be served in period p, where $p > 1$, can be selected later at the end of period $p - 1$ in a *wait-and-see* fashion, using observations of the uncertainty realized

8 Note how this is equivalent to an instance where the items are the elements of S, albeit with weights $q_i \xi_i$, $i \in S$.

up to the time of optimization.[9] In other words, order assignment decisions for periods $p \in P \setminus \{1\}$ are allowed to depend on $\xi^{[p-1]}$ (the customer order realizations up to the previous period), and are obtained through functions $\tilde{S}_p(\cdot)$ that map $\xi^{[p-1]}$ to sets of customer orders to be served in period p. These functions are said to constitute a *robust feasible* solution if, for all possible realizations $\xi \in \Xi$, they evaluate to capacity-feasible order assignments, i.e., if $\tilde{S}_p(\xi)$ can be partitioned into m (possibly empty) capacity-feasible vehicle routes, for each $p > 1$. Amongst all such robust feasible solutions, the decision maker may seek to determine the one that minimizes the cost under a specific realization of future customer orders. In particular, if we select this realization to be the nominal scenario $\hat{\xi}$, we result into the following *multi-stage robust optimization* model:

$$
\begin{aligned}
\underset{\mathbf{S}}{\text{minimize}} \quad & \texttt{CVRP}(S_1, \hat{\xi}, m) + \sum_{p \in P \setminus \{1\}} \texttt{CVRP}(\tilde{S}_p(\hat{\xi}^{[p-1]}), \hat{\xi}, m) \\
\text{subject to} \quad & \tilde{S}_p : \Xi^{[p-1]} \mapsto 2^{\Delta_p} \quad \forall\, p \in P \setminus \{1\} \\
& (S_1, \tilde{S}_2(\xi^{[1]}), \ldots, \tilde{S}_h(\xi^{[h-1]})) \in \mathcal{F} \quad \forall\, \xi \in \Xi \\
& \texttt{BPP}(\tilde{S}_p(\xi^{[p-1]}), \xi) \leq m \quad \forall\, p \in P \setminus \{1\} \quad \forall\, \xi \in \Xi.
\end{aligned}
\tag{$\mathcal{MSRO}$}
$$

Remark 3.2. *Even though decisions implicit in the evaluation of* $\texttt{BPP}(\tilde{S}_p(\xi^{[p-1]}), \xi)$ *are made with full knowledge of the vector of customer order realizations* ξ *over the entire planning horizon, they still satisfy the non-anticipativity requirement. This is because* $\tilde{S}_p(\xi^{[p-1]}) \subseteq \Delta_p$ *and, hence, can only contain orders from* $\bigcup_{q=0}^{p-1} V_q$*. Therefore, the wait-and-see decisions implicit in* $\texttt{BPP}(\tilde{S}_p(\xi^{[p-1]}), \xi)$ *can only depend on customer order realizations up to period* $p-1$*,* $\xi^{[p-1]}$*, for each* $\xi \in \Xi$*.*

Given the computational challenges for the numerical solution of problem $\mathcal{MSRO}$ stemming from the discrete nature of the functional variables $\tilde{S}_p(\cdot)$ and the non-anticipativity requirement across multiple periods, in the following, we propose models to bound $\mathcal{MSRO}$ from above and below via conservative and progressive approximations, respectively.

3.2.3 *Two-Stage Conservative Approximation*

This section presents a non-anticipative, two-stage approximation of $\mathcal{MSRO}$. In this model, the goal is to pre-select the set of (pending as well as potential) customer orders that will be served in *each period* of the planning horizon, irrespectively of whether the potential customer orders will actually be placed or not. In other words, the decision to serve the subset S_p of customer orders, in each future period $p \in P$, is made in a here-and-now fashion, whereas the feasibility of the selected order subsets can be verified

9 Note how this process obeys the *non-anticipativity principle*, which is required for the resulting solution to be implementable in practice.

later in a wait-and-see fashion. As a result, the bin-packing decisions associated with verifying the feasibility of the selected order assignments are allowed to depend on the actual customer order realizations. Consequently, an assignment of orders to periods $\mathbf{S} \in \mathcal{F}$ is said to be robust feasible in this two-stage model if, for all customer order realizations $\xi \in \Xi$ and all periods $p \in P$, the subset of customer orders selected to be served in period p, S_p, can be partitioned into m capacity-feasible vehicle routes. By a similar argument as in Remark 3.2, this partition of S_p can only depend on $\xi^{[p-1]}$, for each $\xi \in \Xi$. Therefore, solutions determined in this manner are non-anticipative by construction, and they can be implemented in practice. The relevant upper-bounding *two-stage robust optimization* model can be cast as follows:

$$\begin{aligned} \underset{\mathbf{S}}{\text{minimize}} \quad & \sum_{p \in P} \texttt{CVRP}(S_p, \hat{\xi}, m) \\ \text{subject to} \quad & (S_1, \ldots, S_h) \in \mathcal{F} \\ & \texttt{BPP}(S_p, \xi) \leq m \quad \forall\, p \in P \setminus \{1\},\ \forall\, \xi \in \Xi. \end{aligned} \tag{$\overline{\mathcal{TSRO}}$}$$

3.2.4 *Two-Stage Progressive Approximation*

This section presents an anticipative, two-stage approximation of $\mathcal{MSRO}$. Similar to the multi-stage model, the goal is to select the set of customer orders to be served in period 1 in a here-and-now fashion, whereas the set of customer orders to be served in period p, where $p > 1$, can be selected later in a wait-and-see fashion. However, in contrast to the multi-stage model, the order assignment decisions for periods $p \in P \setminus \{1\}$ are allowed to depend on the *entire vector* of future customer order realizations ξ (not just on the order realizations up to the previous period, $\xi^{[p-1]}$), and are obtained through functions $\tilde{S}_p(\cdot)$ that map ξ to subsets of customer orders to be served in period p. Consequently, solutions determined in this manner are anticipative by construction. The relevant lower-bounding *two-stage robust optimization* model can be cast as follows:

$$\begin{aligned} \underset{\mathbf{S}}{\text{minimize}} \quad & \texttt{CVRP}(S_1, \hat{\xi}, m) + \sum_{p \in P \setminus \{1\}} \texttt{CVRP}(\tilde{S}_p(\hat{\xi}), \hat{\xi}, m) \\ \text{subject to} \quad & \tilde{S}_p : \Xi \mapsto 2^{\Delta_p} \quad \forall\, p \in P \setminus \{1\} \\ & (S_1, \tilde{S}_2(\xi), \ldots, \tilde{S}_h(\xi)) \in \mathcal{F} \quad \forall\, \xi \in \Xi \\ & \texttt{BPP}(\tilde{S}_p(\xi), \xi) \leq m \quad \forall\, p \in P \setminus \{1\} \quad \forall\, \xi \in \Xi. \end{aligned} \tag{$\underline{\mathcal{TSRO}}$}$$

3.2.5 *Relationship between Two-Stage and Multi-Stage Models*

As already alluded, the optimal value of model $\overline{\mathcal{TSRO}}$ provides a conservative approximation (upper bound) to the optimal value of model $\mathcal{MSRO}$, while the optimal value of model $\underline{\mathcal{TSRO}}$ provides a progressive approximation (lower bound). This is formalized in Proposition 3.1.

Proposition 3.1. *For any uncertainty set Ξ, let $\texttt{DET}(\hat{\xi})$, $\underline{\texttt{TSRO}}$, $\texttt{MSRO}$ and $\overline{\texttt{TSRO}}$ denote the optimal objective values of problems $\mathcal{DET}(\hat{\xi})$, $\underline{\mathcal{TSRO}}$, $\mathcal{MSRO}$ and $\overline{\mathcal{TSRO}}$, respectively. Then, we have*

$$0 \leq \texttt{DET}(\hat{\xi}) \leq \underline{\texttt{TSRO}} \leq \texttt{MSRO} \leq \overline{\texttt{TSRO}}. \tag{3.6}$$

Therefore, the approximation gap of $\overline{\mathcal{TSRO}}$ with respect to $\mathcal{MSRO}$ can be upper bounded as follows:

$$0 \leq \frac{\overline{\texttt{TSRO}} - \texttt{MSRO}}{\texttt{MSRO}} \leq \frac{\overline{\texttt{TSRO}} - \underline{\texttt{TSRO}}}{\underline{\texttt{TSRO}}}. \tag{3.7}$$

Although both $\overline{\mathcal{TSRO}}$ and $\underline{\mathcal{TSRO}}$ are two-stage models, their key difference is that the former is non-anticipative and provides a *causal* policy which only relies on information observed up to the respective day when the solution is to be implemented. The latter model lacks this property and, thus, the customer assignments $\tilde{S}_p(\xi)$ may potentially be selected using future knowledge of customer order realizations. The consequence of assuming future knowledge is that the solutions determined by the model $\underline{\mathcal{TSRO}}$ may become infeasible during actual implementation. We remark, however, that the ability to obtain a valid lower bound using model $\underline{\mathcal{TSRO}}$ is valuable inasmuch as it allows us to establish an upper limit on the potential loss of approximation provided by the non-anticipative model $\overline{\mathcal{TSRO}}$. This is made further evident by the fact that the inequalities in Proposition 3.1 are strict, in general, making the conservatism of a solution an important issue. Nevertheless, as Propositions 3.2 and 3.3 show, there still exist some settings when the two-stage models $\underline{\mathcal{TSRO}}$ and $\overline{\mathcal{TSRO}}$ approximate the multi-stage model $\mathcal{MSRO}$ well.

Proposition 3.2. *For any uncertainty set Ξ, we have $\underline{\texttt{TSRO}} = \texttt{MSRO} = \overline{\texttt{TSRO}}$, whenever any of the following conditions hold:*

(i) The planning horizon spans two time periods, i.e., $h = 2$.

(ii) All customer orders $v \in V$ satisfy (a) $|P_v| \leq 2$, and (b) $1 \in P_v$, if $|P_v| = 2$.

Proposition 3.3. *If problem $\overline{\mathcal{TSRO}}$ is infeasible, then so is $\mathcal{MSRO}$, whenever any of the conditions listed in Proposition 3.2 or any of the following conditions hold.*

(i) The uncertainty set is a hypercube, i.e., $\Xi = \left\{\xi \in \{0,1\}^{|V|} : \xi^0 = \mathbf{1}\right\}$, where $\mathbf{1} \in \mathbb{R}^{|V|}$ denotes the vector of ones.

(ii) The uncertainty set is stage-wise rectangular and disjoint budgeted, i.e., $\Xi = \left\{\xi \in \{0,1\}^{|V|} : \xi^0 = \mathbf{1},\ \sum_{i \in C_{lp}} \xi_{ip} \leq b_{lp},\ \ \forall l \in \{1,\ldots,L_p\}, \forall p \in P\right\}$, where for each $p \in P$, we have $L_p \in \mathbb{Z}_+$, $C_{lp} \subseteq N$ and $C_{lp} \cap C_{l'p} = \emptyset$ for $l \neq l'$; and, all customers $i \in N$ with $w_i \geq 2$ satisfy $\ell_i \geq h$, while all customers $i \in N$ with $w_i = 1$ satisfy $e_i = 1$.

The settings referenced in Propositions 3.2 and 3.3 correspond to cases where customers don't have much flexibility in their day windows. For instance, condition *(ii)*

of Proposition 3.3 states that the two-stage model $\overline{\mathcal{TSRO}}$ is a good approximation of the multi-stage model $\mathcal{MSRO}$, if "flexible" customers (those with at least two feasible service days, $w_i \geq 2$) request service sufficiently in advance of the end of their day window ($\ell_i \geq h$), while "inflexible" customers (those with exactly one feasible service day, $w_i = 1$) request service only one day in advance ($e_i = 1$). Moreover, it can be shown that these results may fail to hold if the conditions stated in the above propositions deviate only slightly. We do not present the relevant counterexamples for the sake of brevity.

It must be mentioned that all four models provide an order assignment for the next day, i.e, day 1 (assuming the model has a feasible solution). We also note that, since each of these models ignores information beyond the planning horizon, none of them can guarantee feasibility in a *rolling horizon* context, in which they will be used (see Section 3.6.5). However, it should be highlighted that the solutions determined by models $\mathcal{DET}(\hat{\xi})$ and $\underline{\mathcal{TSRO}}$ can also become infeasible in a *folding horizon* context, in which solutions are determined through models that are instantiated on successively smaller subsets of the planning horizon and updated to reflect the actual realization of the uncertainty. This is in contrast to model $\overline{\mathcal{TSRO}}$, which provides guarantees of robust feasibility in this manner, and the example in Section 3.3 also illustrates this point.

Finally, we remark that, unlike the multi-stage model, whose numerical solution is challenging to compute, we can develop efficient numerical schemes for models $\overline{\mathcal{TSRO}}$ and $\underline{\mathcal{TSRO}}$. In the following sections, we will provide such schemes and use them to quantify the gap between their objective values for a wide range of problem instances in our numerical experiments.

3.3 EXAMPLE ILLUSTRATING THE DECISION DYNAMICS OF THE DETERMINISTIC AND TWO-STAGE ROBUST OPTIMIZATION MODELS

Consider the instance shown in Figure 3.1 with $n = 6$ customers and $m = 1$ vehicle of capacity $Q = 4$. Note that, for the sake of simplicity, every order from a given customer $i \in N$ always has the same demand q_i and day windows $\{d + e_i, \ldots, d + \ell_i\}$ (assuming the order was placed on day d).

Suppose that the set of pending orders at the end of the current day, i.e., day $0 \in \Pi$, is $V_0 = \{(1, -3), (2, -1), (3, 0), (4, -2)\}$; that is, customers 1, 2 and 4 have respectively placed orders on days -3, -1 and -2 that are still pending, while customer 3 has just placed an order on day 0. There are currently no pending orders from customers 5 and 6. Suppose that there are $h = 4$ days in the planning horizon, and assume that every customer can potentially place an order on any day of the planning horizon. Assume also that we are given a budgeted uncertainty set stipulating that each customer will

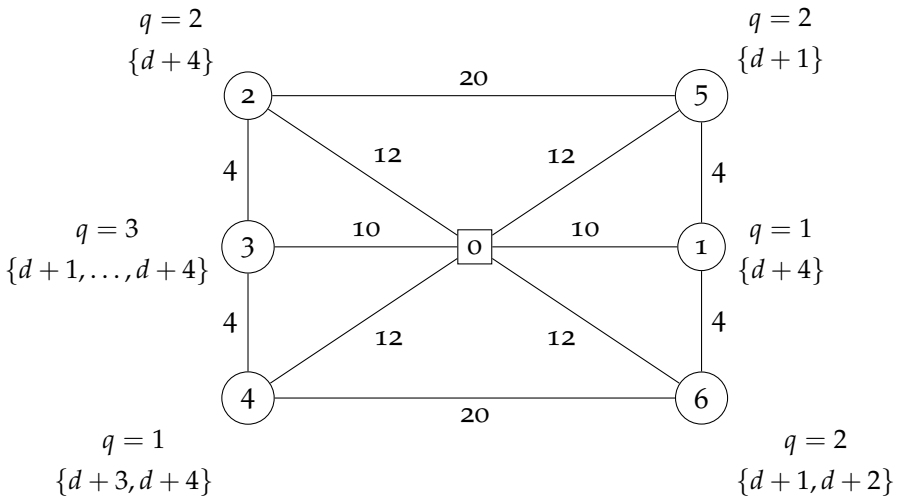


Figure 3.1: Example instance. The number q next to customer i denotes its demand, while the curly braces denote the day window associated with a service request placed on day d. The number above edge (i, j) denotes its transportation cost, c_{ij}. Note that node triplets $(1, 0, 3)$, $(2, 0, 6)$ and $(4, 0, 5)$ are collinear; therefore, the direct connections $(1, 3)$, $(2, 6)$ and $(4, 5)$ exist and have cost $c_{13} = c_{26} = c_{45} = 20$. The edges $(1, 2)$, $(1, 4)$, $(3, 5)$ and $(3, 6)$, which are not shown in this figure, are also assumed to exist and have cost $c_{12} = c_{14} = c_{35} = c_{36} = 18$. Any remaining edges are assumed to not exist, i.e., direct connection between them is not possible.

place at most one order during the planning horizon, and additionally, that no more than a single order will be placed on any given day of the horizon:

$$\Xi = \left\{ \xi \in \{0,1\}^{|V|} : \xi^0 = \mathbf{1},\ \ \xi_{51} + \xi_{52} + \xi_{53} \leq 1,\ \xi_{61} + \xi_{62} \leq 1,\ \xi_{5p} + \xi_{6p} \leq 1\ \forall p \in \{1,2\} \right\}.$$

Observe that potential orders from customers 1–4 as well as a potential order from customer 6 on day 3 are not considered, since their day windows would fall outside the current planning horizon.

ORDERING OF OPTIMAL OBJECTIVE VALUES: Upon solving the example instance using the corresponding models, we obtain the optimal objective values of $\texttt{DET}(\hat{\xi}) = 70$, $\underline{\texttt{TSRO}} = 84$, and $\overline{\texttt{TSRO}} = 88$. This demonstrates that the inequalities (3.6) are strict, in general. The optimal solutions to these models are depicted in Figure 3.2. The structure of the solutions can be intuitively understood as follows. The deterministic model $\mathcal{DET}(\hat{\xi})$ ignores uncertainty and assigns the pending orders in a way that optimizes the cost of routing them; only 1 unit of demand is served on day 1. The two-stage model $\underline{\mathcal{TSRO}}$ considers uncertainty (albeit in an anticipative manner) and serves 2 units of demand on day 1; this frees up some of the vehicle capacity in future time periods, but incurs higher routing cost. The non-anticipative two-stage model $\overline{\mathcal{TSRO}}$ frees up even more of the

vehicle capacity in future time periods, serving 3 units of demand on day 1, but incurs the highest routing cost.

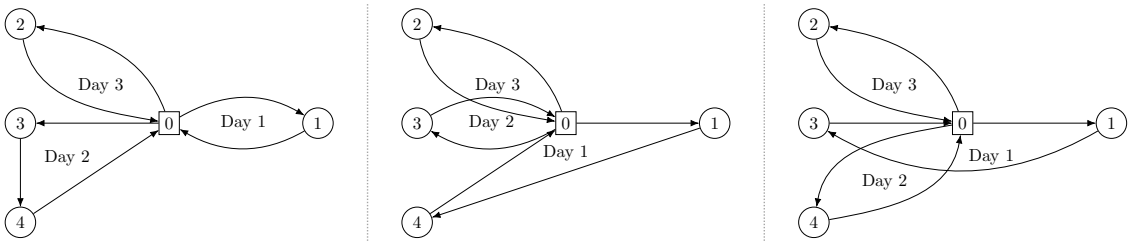


Figure 3.2: From left to right, the graphs illustrate the optimal routes passing through the set of pending orders, corresponding to models $\mathcal{DET}(\hat{\xi})$, $\underline{\mathcal{TSRO}}$ and $\overline{\mathcal{TSRO}}$, respectively. In case of $\underline{\mathcal{TSRO}}$, the routes depict the evaluation of the optimal policy under the nominal scenario, $\hat{\xi}$. Note that, in all these solutions, no pending order is assigned to be served on day 4.

FOLDING HORIZON SCHEME: The folding horizon scheme can be described as follows: we implement only the routes of day 1, and then we re-solve the resulting $(h-1)$-period problem using the model of interest; in doing so, the model is updated to reflect the realization of customer orders that were received at the end of day 1. This process is then continued until the last time period. Figures 3.3, 3.4 and 3.5 respectively illustrate the performance of the solutions determined by models $\mathcal{DET}(\hat{\xi})$, $\underline{\mathcal{TSRO}}$ and $\overline{\mathcal{TSRO}}$ in a folding horizon scheme. In these figures, for each day p of the planning horizon, shaded nodes denote new orders received during day p, solid arrows denote planned routes serving pending orders (as in Figure 2), while patterned nodes and dashed arrows represent pending orders served and routes executed in the past, respectively.

Figure 3.3 shows that the solution determined by the deterministic model becomes infeasible under the realization in which customers 6 and 5 respectively place orders on days 1 and 2.

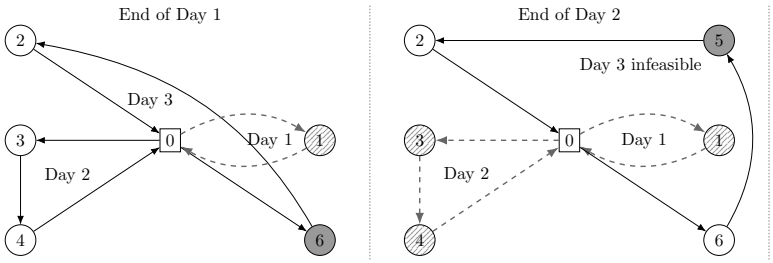
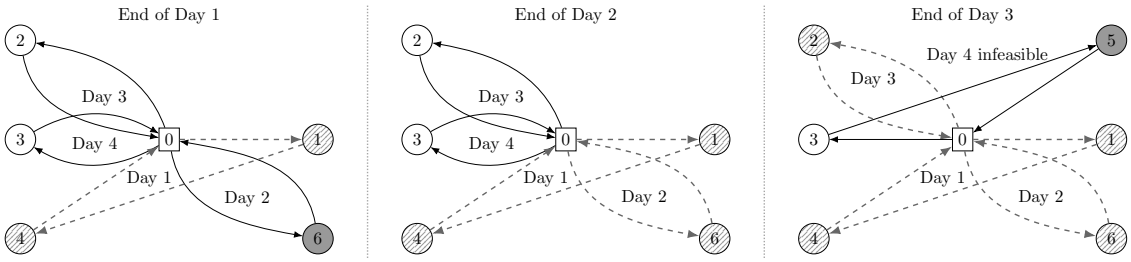
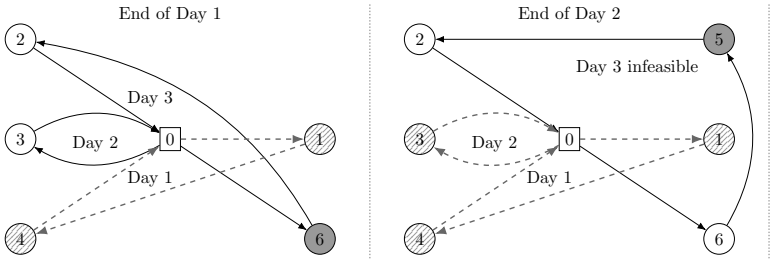


Figure 3.3: Implementation of solutions determined by the deterministic model in a folding horizon scheme. The instance becomes infeasible under the scenario where $\xi_{61} = \xi_{52} = 1$.

Similarly, Figure 3.4 shows that when the route determined by the anticipative two-stage model $\underline{\mathcal{TSRO}}$ is implemented on day 1, the resulting model to be solved at the end of day 1 becomes robust infeasible if customer 6 places an order on day 1. To illustrate this, note that the service day window of this newly received order is $\{2,3\}$, and hence, the order must be served on either day 2 or day 3. If the former were to be chosen, then Figure 3.4a shows that the instance becomes infeasible under the realization in which customer 5 places an order on day 3. Moreover, if the latter were to be chosen, then Figure 3.4b shows that the instance becomes infeasible under the realization in which customer 5 places an order on day 2, and the same is true for the the equivalent solution in which the order from customer 3 is assigned on day 4.



(a) The order received from customer 6 on day 1 is served on day 2. The instance becomes infeasible under the scenario where $\xi_{53} = 1$.



(b) The order received from customer 6 on day 1 is not served on day 2. The instance becomes infeasible under the scenario where $\xi_{52} = 1$.

Figure 3.4: Implementation of solutions determined by the anticipative two-stage model $\underline{\mathcal{TSRO}}$ in a folding horizon scheme.

In contrast to the above, the solution to the non-anticipative two-stage model $\overline{\mathcal{TSRO}}$ remains robust feasible in the context of the folding horizon. Pending orders from customers 1 and 3 are served on day 1. It is then trivial to verify that the customer order assignment $S_2 = \{(4,-2),(5,1),(6,1)\}$, $S_3 = \{(2,-1),(5,2)\}$ and $S_4 = \{(5,3),(6,2)\}$ remain capacity-feasible under every realization admitted by the uncertainty set. Figure 3.5 illustrates this for the particular realization in which customers 6 and 5 respectively place orders on days 1 and 2.

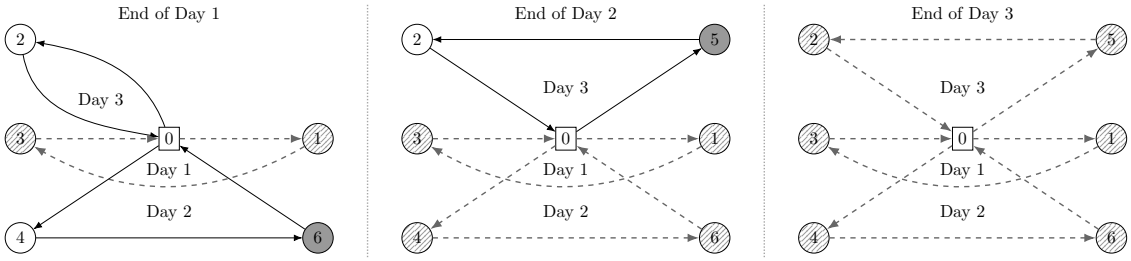


Figure 3.5: Implementation of solutions determined by the non-anticipative two-stage model $\overline{\mathcal{TSRO}}$ in a folding horizon scheme. The depicted order assignments would be implemented under the scenario where $\xi_{61} = \xi_{52} = 1$.

3.4 SOLUTION METHOD

In the previous section, we characterized the solutions to problem $\overline{\mathcal{TSRO}}$ by the subset of customer orders selected to be served on any given day. In practice, however, we shall use mathematical models to describe these solutions through integer variables and numerical methods to obtain their optimal values. Section 3.4.1 describes an integer programing formulation of model $\overline{\mathcal{TSRO}}$, while Section 3.4.2 elaborates on a branch-and-cut framework for its numerical solution.

3.4.1 *Mathematical Formulation*

Recall that a solution to model $\overline{\mathcal{TSRO}}$ is an assignment of customer orders to days. In the following, we shall describe these assignments through binary variables y_{vp} that record whether a customer order $v \in V$ is selected to be served in period $p \in P$. In particular, these variables will enforce the following definition:

$$y_{vp} = 1 \iff v \in S_p \tag{3.8}$$

Note that any feasible assignment $\mathbf{S} \in \mathcal{F}$ induces unique values for y_{vp} through this relationship. Conversely, whenever the binary variables y_{vp}, $v \in V$, $p \in P$ satisfy the equation

$$\sum_{p \in P} y_{vp} = \sum_{p \in P_v} y_{vp} = 1 \quad \forall\, v \in V, \tag{3.9}$$

requiring each customer order to be served exactly once during the planning horizon within its day window, the values of y_{vp} also induce a unique assignment of customer orders to days, and we shall denote this assignment by $\mathbf{S}(y)$.

In order to evaluate the cost of the solutions under scenario $\hat{\xi}$, we will use integer variables x_{uvp} to indicate whether a vehicle serves order $v \in V_0$ immediately after order u (or the depot 0) in period $p \in P$. To simplify notation, we define

$$E_0 = \left\{ \left((i,d),(j,d')\right) \in V_0 \times V_0 : (i,j) \in E \text{ or } i = j, d \neq d' \right\}$$

as the subset of edges which cover the pending customer orders and over which it is sufficient to define routing variables x in order to evaluate the objective function of model $\overline{\mathcal{TSRO}}$. Furthermore, given a set of customer orders $S \subseteq V_0$, we define $E_0(S) = \{(u,v) \in E_0 : u,v \in S, u \neq v\}$ as the set of edges that connect orders in S. Following standard vehicle routing modeling techniques, we derive the following integer programming formulation that is valid for the deterministic model under the nominal scenario $\mathcal{DET}(\hat{\xi})$. We will use it as a basis in order to derive a valid formulation for $\overline{\mathcal{TSRO}}$.

$$\underset{x,y}{\text{minimize}} \quad \sum_{p\in P}\sum_{v\in V_0} c_{0v}x_{0vp} + \sum_{p\in P}\sum_{(u,v)\in E_0} c_{uv}x_{uvp} \tag{3.10a}$$

$$\text{subject to} \quad y_{vp} \in \{0,1\} \quad \forall\, v \in V,\ \forall\, p \in P \tag{3.10b}$$

$$x_{0vp} \in \{0,1,2\} \quad \forall\, v \in V_0,\ \forall\, p \in P \tag{3.10c}$$

$$x_{uvp} \in \{0,1\} \quad \forall\, (u,v) \in E_0,\ \forall\, p \in P \tag{3.10d}$$

$$\sum_{p\in P} y_{vp} = \sum_{p\in P_v} y_{vp} = 1 \quad \forall\, v \in V \tag{3.10e}$$

$$\sum_{v\in V_0} x_{0vp} \leq 2m \quad \forall\, p \in P \tag{3.10f}$$

$$x_{0vp} + \sum_{u:(u,v)\in E_0} x_{uvp} = 2y_{vp} \quad \forall\, v \in V_0,\ \forall\, p \in P \tag{3.10g}$$

$$\sum_{p\in P}\sum_{(u,v)\in E_0(S)} x_{uvp} \leq |S| - \left\lceil \frac{1}{Q}\sum_{i\in S} q_i \right\rceil \quad \forall\, S \subseteq V_0. \tag{3.10h}$$

The objective (3.10a) consists of minimizing the total cost of routing the pending orders V_0 across the planning horizon; equations (3.10e) stipulate that each known customer order must be served exactly once within its day window; constraints (3.10f) require that no more than m vehicles depart from the depot on any given day; equations (3.10g) state that, if customer order $v \in V_0$ is served on day $p \in P$, then there must be exactly two edges incident to v on day p. Finally, constraints (3.10h) restrict subtours, requiring that each customer order is served by a vehicle that departs from and returns to the depot, as well as enforce the vehicle capacity restrictions by imposing applicable lower bounds on the number of vehicles that serve a set of orders $S \subseteq V_0$.

In order for the binary variables y_{vp} in the above formulation to induce a customer order assignment $\mathbf{S}(y)$ that is robust feasible in $\overline{\mathcal{TSRO}}$ we must be able to serve $S_p(y)$ using at most m vehicles. We show that the following *robust cover inequalities* characterize the subsets of customer orders that can be served in any period $p \in P$, $S_p(y)$, such that $\mathbf{S}(y)$ is robust feasible in $\overline{\mathcal{TSRO}}$:

$$m + \sum_{v\in S}\left(1 - y_{vp}\right) \geq \texttt{BPP}(S,\xi) \quad \forall\, S \subseteq V,\ \forall\, p \in P,\ \forall\, \xi \in \Xi. \tag{3.11}$$

Proposition 3.4. *For any support Ξ and binary variables y_{vp}, $v \in V$, $p \in P$, we have that*

1. *the robust cover inequalities (3.11) are* necessary *to induce a robust feasible customer order assignment $\mathbf{S}(y)$ in $\overline{\mathcal{TSRO}}$; and*

2. *in conjunction with equations* (3.9)*, the robust cover inequalities* (3.11) *are* sufficient *to induce a robust feasible customer order assignment* $\mathbf{S}(y)$ *in* $\overline{\mathcal{TSRO}}$*.*

Constraints (3.11) require that for every set $S \subseteq V$ that is a subset of the customer orders selected to be served in period p, the bin packing value associated with S under any realization $\xi \in \Xi$ must be no more than the number of vehicles available. For any candidate assignment $S_p(y)$, the left-hand side of the constraint associated with period p and $S = S_p(y)$ evaluates to m, which implies that there exists a capacity-feasible partition of $S_p(y)$ into at most m components, for every realization $\xi \in \Xi$. The formulation for $\overline{\mathcal{TSRO}}$, which we shall denote by $\overline{\mathcal{TSRO}}_{IP}$, is obtained by appending constraints (3.11) to formulation (3.10).

3.4.2 *Branch-and-Cut Framework*

We use a branch-and-cut algorithm to solve formulation $\overline{\mathcal{TSRO}}_{IP}$. A branch-and-cut algorithm embeds the addition of *cutting planes* within each tree node of a branch-and-bound algorithm. The solution at the root node is obtained by solving the linear programming (LP) relaxation consisting only of constraints (3.10e)–(3.10g) along with the variable bounds. Since the number of constraints (3.10h) and (3.11) is exponential in the size of the instance, we remove these inequalities and treat them in a cutting plane fashion by dynamically reintroducing them whenever the node solution is found to violate them. Section 3.4.3 describes other families of inequalities that are valid for the convex hull of integer feasible solutions of formulation $\overline{\mathcal{TSRO}}_{IP}$. Unlike constraints (3.10h) and (3.11), these inequalities are not necessary to characterize the set of integer feasible solutions of our formulation; however, they are capable of strengthening the LP relaxation in each node and, therefore, can be used as cutting planes in order to expedite the search process.

Section 3.4.4 describes algorithms for solving the separation problems for inequalities (3.10h), (3.11) and the other families of inequalities described in Section 3.4.3. When violated inequalities are identified, the node solution is re-computed by adding all violated inequalities to the LP relaxation and this procedure is iterated until no new inequalities are generated. If the final node solution happens to satisfy all of the integrality constraints (3.10b)–(3.10d), then it is accepted as the new *incumbent* solution since, in such cases, our algorithms for solving the separation problems for inequalities (3.10h) and (3.11) are exact (i.e., they provide guarantees to identify a violating member, if one exists). Otherwise, new sub-problems (i.e., nodes) are created by branching on an integer variable whose value in the current node solution is fractional. The results in Section 3.6 have been obtained by using the default branching strategy provided by the solver. Finally, since all identified inequalities are valid globally (i.e., for all nodes of the branch-and-bound tree), we add them to the LP relaxation of each open node of the tree.

3.4.3 *Valid Inequalities*

3.4.3.1 *Lifted Robust Cover Inequalities*

Observe that, if the right-hand side of the robust cover inequality (3.11) is not greater than m, then it is dominated by the trivial variable bounds: $0 \leq y_{vp} \leq 1$. On the other hand, if its right-hand side is strictly greater than m, then the following proposition shows that it is possible to lift the resulting robust cover inequality. This lifting result is analogous to the lifting of valid cover inequalities for the $0-1$ knapsack polytope to the so-called *extended cover inequalities* (see [24]). In this proposition, $C(S) := \{v \in V \setminus S : q_v \geq \max\{q_j : j \in S\}\}$ denotes the set of all customer orders with higher demand than any order in $S \subseteq V$.

Proposition 3.5. *For a customer order subset $S \subseteq V$, suppose that $\texttt{BPP}(S, \tilde{\xi}) \geq m + k$ is satisfied for some $\tilde{\xi} \in \Xi$, where $k \in \mathbb{N}$. Then, the following inequality is valid for formulation $\overline{\mathcal{TSRO}}_{IP}$:*

$$\sum_{v \in S} \tilde{\xi}_v \left(1 - y_{vp}\right) - \sum_{v \in C(S)} \tilde{\xi}_v y_{vp} \geq k \quad \forall\, p \in P. \tag{3.12}$$

3.4.3.2 *Robust Cumulative Capacity Inequalities*

These inequalities enforce that the cumulative demand to be served in period $p \in P$, under any customer order realization $\xi \in \Xi$, does not exceed the total fleet capacity mQ. Unlike the robust cover inequalities (3.11), they ignore bin packing considerations and do not guarantee that the set of customer orders selected to be served in period p can be packed into the available fleet. Nevertheless, it can be shown that they do not dominate and are not dominated by the robust cover inequalities (3.11); that is, node solutions of the branch-and-bound tree may violate them without violating inequalities (3.11) and vice versa.

$$\sum_{v \in V} q_v \xi_v y_{vp} \leq mQ \quad \forall\, p \in P,\ \forall\, \xi \in \Xi. \tag{3.13}$$

3.4.3.3 *Valid Inequalities from CVRP*

The *two-index vehicle flow formulation* is one of the most popular formulations for the CVRP. In this formulation, integer variables x'_{ij} count the number of times edge (i, j) is traversed by any vehicle in a solution of the CVRP. Several families of inequalities are known to be valid for the corresponding convex hull of integer feasible solutions, including the so-called *rounded capacity*, *framed capacity*, *strengthened comb*, *multistar* and *hypotour inequalities*, among others (see [193]). The following proposition shows that any such inequality can also be made valid for formulation (3.10) by disaggregating (across periods) the corresponding edge variables that appear in the inequality.

Proposition 3.6. *Let $\sum_{(i,j) \in I} \lambda_{ij} x'_{ij} \leq \mu$ be any inequality that is valid for the two-index formulation of the CVRP instance defined on the subgraph of G with depot node 0, customers*

V_0, demands q_v, $v \in V_0$ and vehicle capacity Q. Then, $\sum_{p\in P}\sum_{(i,j)\in I}\lambda_{ij}x_{ijp} \leq \mu$ is valid for formulation (3.10).

Apart from the above inequalities, the following *generalized subtour elimination constraints* (3.14) and *generalized fractional capacity inequalities* (3.15) are also valid for formulation (3.10). These inequalities enforce lower bounds on the number of edges between $S \subseteq V_0$ and its complement in time period $p \in P$, whenever at least one member of S is served in period p. It can be shown that they do not dominate and are not dominated by constraints (3.10h).

$$\sum_{(u,v)\in E_0(S)} x_{uvp} \leq |S| - y_{vp} \quad \forall\, S \subseteq V_0,\ \forall\, v \in S,\ \forall\, p \in P. \tag{3.14}$$

$$\sum_{(u,v)\in E_0(S)} x_{uvp} \leq |S| - \left(\frac{1}{Q}\sum_{v\in S} q_v y_{vp}\right) \quad \forall\, S \subseteq V_0,\ \forall\, p \in P. \tag{3.15}$$

3.4.4 *Separation Algorithms*

3.4.4.1 *Robust Cover Inequalities*

Since the bin packing problem is NP-hard, the separation problem for inequalities (3.11) is also NP-hard, even if Ξ consists of a single element. This motivates the need for separation heuristics. Nevertheless, as mentioned previously, in the context of a branch-and-cut algorithm, the procedure to solve these separation problems must be exact if the current node solution satisfies all of the integrality constraints (3.10b)–(3.10d). Otherwise, a heuristic procedure to identify violated inequalities suffices. In the following, we describe procedures to solve the associated separation problems separately for the cases when the node solution is integral (i.e., satisfies (3.10b)–(3.10d)) or fractional. For the remainder of this section, we shall assume that (x^*, y^*) is the current node solution for which we want to identify violated robust cover inequalities.

We remark that we do not explicitly separate the lifted robust cover inequalities (3.12). Instead, we use Proposition 3.5 to lift any identified violating member of (3.11) and add the lifted form of the inequality to the current node solution.

FRACTIONAL NODE SOLUTIONS: Typically, fractional node solutions are encountered much more frequently in the branch-and-bound tree than those with integral solutions. Therefore, the separation procedures employed at such nodes must be computationally efficient, although not necessarily exact. In this context, note that inequality (3.11) remains valid if we replace its right-hand side with a lower bound. In the following, we attempt to separate the following relaxed version of the robust cover inequalities (3.11), obtained by replacing the optimal value of the bin packing problem with the so-called L_1 lower

bound [196]. We remark that, although the L_1 bound of the bin packing problem may deviate from its optimal value by up to 50% in the worst case, it is typically tight when the item weights are sufficiently small with respect to the bin capacity.

$$m + \sum_{v \in S} (1 - y_{vp}) \geq \left\lceil \frac{1}{Q} \sum_{v \in S} q_v \xi_v \right\rceil \quad \forall\, S \subseteq V,\ \forall\, p \in P,\ \forall\, \xi \in \Xi. \tag{3.11'}$$

Observe that there always exists a maximally violating member of the family of inequalities (3.11′) satisfying $\xi_v = 1$ for all $v \in S$. Indeed, consider a member of (3.11′) defined by $S \subseteq V$, $p \in P$ and $\xi \in \Xi$ such that $\xi_v = 0$ holds for some $v \in S$. Then, we can obtain a more violated member defined by the same p and ξ but considering the subset $S \setminus \{v\}$: the right-hand side of this inequality is the same as that of S, while its left-hand side can only decrease with respect to S. With this observation, the separation problem for inequalities (3.11′), for a given $p \in P$, can be formulated as the following binary program, where the variable $z_v \in \{0, 1\}$ indicates whether $v \in S$.

$$\underset{z \in \{0,1\}^{|V|}, \xi \in \Xi}{\text{minimize}} \left\{ \sum_{v \in V} (1 - y^*_{vp}) z_v : z \leq \xi,\ \sum_{v \in V} q_v z_v \geq mQ + \varepsilon \right\} \tag{3.16}$$

If (3.16) happens to be infeasible, then no violations are possible for the given $p \in P$. Otherwise, if $m + \sum_{v \in S^*} (1 - y^*_{vp}) > \lceil \sum_{v \in S^*} q_v / Q \rceil$ is satisfied, where $S^* = \{v \in V : z^*_v = 1\}$ is defined by the optimal solution (z^*, ξ^*) of (3.16), then the member of (3.11′) defined by S^* and realization ξ^* corresponds to a most violated inequality. Note that, without loss of optimality, we can fix to zero all variables z_v such that $y^*_{vp} = 0$. This is because $z_v = 1$ implies that the violation of the resulting inequality corresponding to S^* would only decrease with respect to that corresponding to $S^* \setminus \{v\}$.

Observe that, for fixed $\xi \in \Xi$, (3.16) reduces to a standard knapsack problem, which typically can be solved very efficiently by means of specialized algorithms [196]. In any case, our computational experience with the instances solved in Section 3.6 suggested that the optimal solution of problem (3.16) can be obtained very easily by using a commercial integer programming solver, and the results in that section were therefore obtained in this manner.

INTEGRAL NODE SOLUTIONS: For each $p \in P$, $S_p(y^*) = \{v \in V : y^*_{vp} = 1\}$ is the candidate set of orders selected to be served in period p in the current node solution. In order to identify violated inequalities in period p, it is sufficient to consider the inequality for $S = S_p(y^*)$. This is because: *(i)* for any subset of S, the left-hand side of (3.11) evaluated at the current node solution remains the same as for S, while the right-hand side can only decrease with respect to that for S; *(ii)* for any set $S \cup \{v\}$, where $v \in V$ is such that $y^*_{vp} = 0$, the left-hand side of (3.11) evaluated at the current node solution increases by one (with respect to S) while the right-hand side increases by at most one. In either case, the magnitude of violation of the resulting inequality can never increase with respect to that for S. Stated differently, the current node solution

maximally violates the member of (3.11) corresponding to $S = S_p(y^*)$, for given $p \in P$ and $\xi \in \Xi$. For this choice of S, the left-hand side of (3.11) evaluates to m and satisfaction of the inequality is equivalent to checking if

$$m \geq \texttt{BPP}(S, \xi^*), \text{where } \xi^* \in \arg\max_{\xi \in \Xi} \texttt{BPP}(S, \xi).$$

If $m < \texttt{BPP}(S, \xi^*)$, then the inequality corresponding to set S, period p and customer order realization ξ^* is added to the current node solution. If no violations are found in any $p \in P$, then the current node solution is guaranteed to satisfy each member of (3.11).

The computation of a maximizer of $\texttt{BPP}(S, \xi)$ needs to be done as often as the branch-and-bound tree encounters integral node solutions and, hence, it is crucial that this can be done efficiently. For general supports Ξ, this requires the solution of a bilevel integer program in which the (upper level) problem is to determine a scenario $\xi \in \Xi$ that maximizes the optimal value of the (lower level) bin packing problem $\texttt{BPP}(S, \xi)$. It is computationally difficult to solve this problem using existing methods (e.g., see [101, 248, 276]), since they typically do not address problems containing bilinear terms between upper and lower level decisions. In order to address this and other limitations, we present in Appendix 3.10 a numerically efficient solution procedure that improves upon and extends the column-and-constraint generation framework [280] for solving such problems. By formulating the lower-level bin packing problem as a feasibility problem, the proposed procedure does not necessarily compute a maximizer of $\texttt{BPP}(S, \xi)$. Instead, it either certifies that $\texttt{BPP}(S, \xi^*) \leq m$, or returns a realization ξ for which $\texttt{BPP}(S, \xi) > m$. In either case, the exactness of the separation procedure is guaranteed.

We remark that the procedure presented above can address any uncertainty set Ξ that satisfies the conditions (C1)–(C3) described in Section 3.2. It turns out that, for specially structured disjoint budget uncertainty sets of the type shown in (3.2), we can compute a maximizer of $\texttt{BPP}(S, \xi)$ more efficiently, avoiding the solution of a bilevel program.

Proposition 3.7. *Assume that the uncertainty set Ξ is of type (3.2) and disjoint budgeted. Also, for any $S \subseteq V$ and $l = 1, \ldots, L$, assume that $v_{l,1}, v_{l,2}, \ldots, v_{l,|S \cap B_l|}$ represents an ordering of the customer orders in the set $S \cap B_l$ according to non-increasing demand; that is, $q_{v_{l,1}} \geq \ldots \geq q_{v_{l,|S \cap B_l|}}$. Let $J_0 := V_0 \cup S \setminus \cup_{l=1}^{L} B_l$; and, for $l = 1, \ldots, L$, let $J_l := \{v_{l,1}, \ldots, v_{l,j_l}\}$, where $j_l = \min\{b_l, |S \cap B_l|\}$. Then, an optimal solution ξ^* of $\max\{\texttt{BPP}(S, \xi) : \xi \in \Xi\}$ is given by*

$$\xi^*_v = \begin{cases} 1, & \text{if } v \in J_0 \cup J_1 \cup \ldots \cup J_L \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } v \in V.$$

With suitably chosen data structures, Proposition 3.7 shows that we can obtain a maximizer ξ^* of the right-hand side of inequality (3.11) for a given $S \subseteq V$ in time $O(|S| + \sum_{l=1}^{L} b_l \log b_l)$ using a partial sorting algorithm. Once the maximizer is obtained, a deterministic bin packing problem $\texttt{BPP}(S, \xi^*)$ must be solved in order to check if a violation exists. In our implementation, we solved these bin packing problems using the exact algorithm MTP described in [196].

3.4.4.2 *Robust Cumulative Capacity Inequalities*

For fixed $p \in P$, the separation problem for the robust cumulative capacity inequalities (3.13) can be formulated as a binary program: $\max\left\{\sum_{v\in V} q_v y^*_{vp}\xi_v : \xi \in \Xi\right\}$. Here, (x^*, y^*) represents the current node solution. If $\sum_{v\in V} q_v y^*_{vp}\xi^*_v > mQ$ is satisfied, where ξ^* is the optimal solution of the binary program, then the inequality (3.13) corresponding to period p and customer order realization ξ^* is violated. As in the case of the robust cover inequalities, for disjoint budget uncertainty sets, the solution of a binary program can be avoided. In such cases, a slight modification of Proposition 3.7 allows us to compute ξ^* in time $O(|V| + \sum_{l=1}^{L} b_l \log b_l)$. Specifically, we apply Proposition 3.7 to the set $S = V$ but, instead of ordering the elements in the set $S \cap B_l = V \cap B_l = B_l$ by non-increasing demand, we order them by non-increasing values of $q_v y^*_{vp}$.

3.4.4.3 *Valid Inequalities from CVRP*

Proposition 3.6 shows that any family of valid inequalities for the CVRP can be made valid for formulation $\overline{\mathcal{TSRO}}_{IP}$. Violated members of these families of inequalities (including (3.10h)) can be identified using the same observation. Specifically, given the current node solution (x^*, y^*), compute the corresponding two-index solution: $x'^*_{ij} = \sum_{p\in P} x^*_{ijp}$ for each (i, j). The resulting vector x'^* can then be used as input to any separation algorithm for the corresponding family of inequalities. In our implementation, we used the CVRPSEP package [193] for separating inequalities (3.10h) along with the framed capacity, comb, multistar and hypotour inequalities. o. The generalized subtour elimination constraints (3.14) can be separated in polynomial time by solving $|V_0|$ maximum flow problems. We refer the reader to [113] for details, wherein a procedure to generate more than one violated inequality is also described; the results in Section 3.6 were obtained using this procedure. The separation problem for the generalized fractional capacity inequalities can also be solved in polynomial time because it reduces to the separation problem for the standard *fractional capacity inequalities* for the CVRP after modifying the customer demands to be $q_v y^*_{vp}$ for each $v \in V_0$. McCormick, Rao, and Rinaldi [197] describe routines for solving the associated separation problems, which we also used in obtaining the results of Section 3.6.

3.5 COMPUTATION OF LOWER BOUNDS

Proposition 3.1 shows that we can bound from below the optimal value of model $\mathcal{MSRO}$ and, hence, of $\overline{\mathcal{TSRO}}$ via the optimal value of $\underline{\mathcal{TSRO}}$. Therefore, using this value, we can quantify how well $\overline{\mathcal{TSRO}}$ approximates $\mathcal{MSRO}$. In this section, we discuss how to numerically compute the optimal value of $\underline{\mathcal{TSRO}}$. The approach is a branch-and-cut algorithm similar to that for solving $\overline{\mathcal{TSRO}}$ and much of the discussion mirrors

Sections 3.4.1 and 3.4.2; therefore, we only emphasize the aspects which differ in the two cases.

3.5.1 *Mathematical Formulation*

Formulation (3.10) serves as the basis of an integer programming model for $\underline{\mathcal{TSRO}}$, which we shall denote by $\underline{\mathcal{TSRO}}_{IP}$. While the interpretation of binary variables y_{vp} was straightforward in $\overline{\mathcal{TSRO}}_{IP}$, in the case of $\underline{\mathcal{TSRO}}_{IP}$, variables y_{vp} will indicate whether customer order $v \in V$ is selected to be served in period $p \in P$ in the customer order assignment obtained by evaluating the optimal solution policy under scenario $\hat{\xi}$:

$$\begin{aligned} &y_{v1} = 1 \iff i \in S_1 \\ &y_{vp} = 1 \iff i \in \tilde{S}_p(\hat{\xi}) \text{ for all } p \in P \setminus \{1\}. \end{aligned} \tag{3.17}$$

Note that any solution $(S_1, \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ of $\underline{\mathcal{TSRO}}$ induces unique values for y_{vp} through this relationship. Conversely, whenever the binary variables y_{vp} satisfy equation (3.9), their values induce a unique assignment on day 1, which we shall denote by $S_1(y)$, that would be common across all feasible customer order assignments $(S_1, \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$. However, rather than construct explicit functional forms of $\tilde{S}_p(\cdot)$, formulation $\underline{\mathcal{TSRO}}_{IP}$ only enforces the existence of one. The existence of a feasible solution in $\underline{\mathcal{TSRO}}$ is equivalent to the existence of feasible assignments for all realizations of the uncertainty, with the additional restriction that they must share the same here-and-now assignment, $S_1(y)$. This, in turn, is equivalent to the existence of a capacity-feasible and day window-feasible partition of the orders $V \setminus S_1(y)$ into $h-1$ subsets, $\tilde{S}_2(\xi), \ldots, \tilde{S}_h(\xi)$, for each $\xi \in \Xi$.

Motivated by this observation, for any $S \subseteq V \setminus \{v \in V : P_v = \{1\}\}$, and for any $\xi \in \Xi$, let $\texttt{BPPDW}(S, \xi)$ denote the optimal value of an instance of the *Bin Packing Problem with Day Windows*. In this problem, the bin size is Q, the set of days is $P \setminus \{1\}$, and the items are the elements of S featuring weights $q_v \xi_v$ and day windows $P_v \setminus \{1\}$ for each $v \in S$. Further, at least m (possibly empty) bins must be used used on each day $p \in P \setminus \{1\}$. The requirement of using at least m bins on each day is necessary to disallow the case where the optimal solution of the bin packing problem uses less than $m(h-1)$ bins overall, but more than m bins on some day. We show that the following robust cover inequalities characterize the sets of here-and-now customer order assignments that can be part of a feasible solution in $\underline{\mathcal{TSRO}}$:

$$m(h-1) + \sum_{v \in S} y_{v1} \geq \texttt{BPPDW}(S, \xi) \quad \forall\, S \subseteq V \setminus \{v \in V : P_v = \{1\}\},\ \forall\, \xi \in \Xi. \tag{3.18}$$

Proposition 3.8. *For any support Ξ and binary variables y_{vp}, $v \in V$, $p \in P$, we have that*

1. *the robust cover inequalities (3.18) are* necessary *to induce a customer order assignment on day 1, $S_1(y)$, that guarantees the existence of a feasible solution $(S_1(y), \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ in $\underline{\mathcal{TSRO}}$.*

2. *in conjunction with equations* (3.9)*, the robust cover inequalities* (3.11) *are* sufficient *to induce a customer order assignment on day* 1*,* $S_1(y)$*, that guarantees the existence of a feasible solution* $(S_1(y), \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ *in* $\underline{\mathcal{TSRO}}$*.*

Constraints (3.18) require that, for every set $S \subseteq V$ that is a subset of the customer orders selected to be served on days other than day 1, the bin packing value associated with S (considering day windows) under any realization $\xi \in \Xi$ must be no more than the total number of vehicles available over the planning horizon. For any assignment $S_1(y)$, the left hand side of the constraint associated with $S = V \setminus S_1(y)$ evaluates to $m(h-1)$, which implies that there exists a capacity feasible partition of $V \setminus S_1(y)$ into $h-1$ solutions, each containing at most m components, for every realization $\xi \in \Xi$. The formulation for $\underline{\mathcal{TSRO}}$ is obtained by adding constraints (3.18) to formulation (3.10).

Remark 3.3. *The Bin Packing Problem with Day Windows can be viewed as a special case of the well-studied* Bin Packing Problem with Conflicts *[161] and may be modeled as such, albeit with an additional restriction that accounts for using at least m bins on each day. Indeed, given any instance of the former, we can construct an instance of the latter by using the same set of items, the same bin size and constructing the so-called "conflict graph" by defining edges* (u, v) *whenever* $P_u \cap P_v = \emptyset$*, for u and v in the set of items. However, without the additional restriction of using at least m bins on each day, the resulting instance of the Bin Packing Problem with Conflicts constitutes only a relaxation of the Bin Packing Problem with Day Windows.*

3.5.2 *Branch-and-Cut Framework*

We use a branch-and-cut algorithm to solve formulation $\underline{\mathcal{TSRO}}_{IP}$. The initial LP relaxation consists of constraints (3.10e)–(3.10g) along with variable bounds. Constraints (3.10h) and (3.18) are removed and are dynamically reintroduced as cutting planes whenever the node solution is found to violate them. The CVRP inequalities described in Section 3.4.3.3 are valid for $\underline{\mathcal{TSRO}}_{IP}$ as well and can be used in the branch-and-cut algorithm as cutting planes. Similarly, the robust cumulative capacity inequalities described in Section 3.4.3.2 are also valid, albeit with a slight modification, as follows.

$$\sum_{v \in V} q_v \xi_v (1 - y_{v1}) \leq m(h-1)\, Q \quad \forall\, \xi \in \Xi. \tag{3.19}$$

These inequalities enforce that the total demand to be served on days other than day 1, under any customer order realization $\xi \in \Xi$, does not exceed the total fleet capacity available on those days. The procedure to solve the separation problem for these inequalities is similar to that described in Section 3.4.4.2. Finally, it can also be shown that they do not dominate and are not dominated by the robust cover inequalities (3.18).

The remainder of this section describes our separation procedures for the robust cover inequalities (3.18) employed within the branch-and-cut algorithm. We shall assume that (x^*, y^*) is the current node solution for which we want to identify violated inequalities. If

the current node solution happens to be fractional, i.e., does not satisfy constraints (3.10b)–(3.10d), then we can relax the robust cover inequality (3.18) and replace its right-hand side with a valid lower bound. For example, the L_1 bound described in Section 3.4.4.1 is a valid lower bound for the bin packing problem with day windows as well, and the procedure described therein can be utilized to solve the separation problems of the corresponding relaxed version of inequalities (3.18). Similarly, Remark 3.3 shows that we can utilize lower bounds to the bin packing problem with conflicts [127] to derive valid relaxations of the robust inequalities (3.18). However, preliminary computational experiments showed that these lower bounds are typically weak and that it does not pay off to expend additional computational effort in separating the relaxed inequalities as they are almost never violated.

On the other hand, if the current node solution happens to be integral, i.e., satisfies constraints (3.10b)–(3.10d), then the separation procedure for the robust cover inequalities (3.18) must be exact. In such cases, $S_1(y^*) = \{v \in V : y^*_{v1} = 1\}$ is the candidate set of customer orders selected to be served on day 1. By a similar argument as in Section 3.4.4.1, it can be shown that the current node solution maximally violates the member of the family of inequalities (3.18) corresponding to $S = V \setminus S_1(y^*)$, for given $\xi \in \Xi$. For this choice of S, the left-hand side of (3.18) evaluates to $m(h-1)$ and satisfaction of the inequality is equivalent to checking if

$$m(h-1) \geq \texttt{BPPDW}(S, \xi^*), \text{where } \xi^* \in \arg\max_{\xi \in \Xi} \texttt{BPPDW}(S, \xi).$$

If $m(h-1) < \texttt{BPPDW}(S, \xi^*)$, then the inequality corresponding to set S and customer order realization ξ^* is violated. The computation of ξ^* requires the solution of a bilevel integer program, for which we use the procedure described in Appendix 3.10. By formulating the bin packing problem as a feasibility problem, the procedure either certifies that $\texttt{BPPDW}(S, \xi^*) \leq m(h-1)$, or returns a realization ξ for which $\texttt{BPPDW}(S, \xi) > m(h-1)$, thus guaranteeing the exactness of the separation procedure.

3.6 COMPUTATIONAL EXPERIMENTS

This section presents computational results obtained using the branch-and-cut algorithms described in Sections 3.4.2 and 3.5.2 on benchmark instances from the literature. Specifically, in Section 3.6.1, we present the characteristics of the test instances; in Section 3.6.2, we present a summary of the numerical performance of our algorithm; in Section 3.6.3, we present detailed tables of results outlining the effect of the various inequalities; in Section 3.6.4, we analyze the approximation quality of our two-stage model; and finally, in Section 3.6.5, we study the performance of the proposed robust optimization model in a rolling horizon simulation framework, and we compare it to the performance of the deterministic model as well as two decision approaches that are popular among practitioners.

The algorithms were implemented in C++ using the C callable library of CPLEX 12.7 and compiled with the GCC 5.1.0 compiler. The experiments were conducted on a single-core of an Intel Xeon 3.1 GHz processor with a software imposed memory limit of 6 GB. In the implementation of the algorithms, all CPLEX-generated cutting planes were disabled because enabling them increased overall computation times; all other solver options were left at their default values. Interested readers can freely download our code implementation, datasets used, and associated user instructions in the following link: http://gounaris.cheme.cmu.edu/codes/mpvrp.

3.6.1 *Test Instances*

Our instances are derived from the standard CVRP benchmark instances in the so-called A, B, E, F, M and P datasets,[10] which are usually adopted to generate benchmark instances for several variants of the vehicle routing problem. The number of customer nodes in these CVRP instances range from 12 to 199. From each of these CVRP instances, we generate a single multi-period VRP instance using the procedure described in Baldacci et al. [32]. Since the work of Baldacci et al. [32] does not study the problem in the context of the longer (possibly infinite) time horizon Π, we interpret the resulting five-period VRP instance (i.e., $h = 5$) as a possible instantiation of the deterministic model to be solved at the end of day 0 of the longer horizon. Specifically, we let V_0 be the set of pending orders as specified in Baldacci et al. [32], and we let N be the set of customers in the original CVRP instance. For each $i \in N$, q_i is set equal to the demand in the original instance; e_i is sampled uniformly in $[\rho_i, h]$, where ρ_i is the first day of the day window $\{\rho_i, \ldots, \delta_i\}$ as specified in Baldacci et al. [32];[11] and, ℓ_i is set to be $e_i + w_i - 1$, where the width w_i is set equal to the original width, $\delta_i - \rho_i + 1$, if $\rho_i > 1$, and is uniformly sampled in $[\delta_i, 3]$ otherwise. In the resulting instance, the average width of a day window ($= w_i$) is 2, and the average number of days elapsed between the day at which an order is placed and the first feasible service day ($= e_i$) is 3.5, which are similar to the corresponding values (2.5 and 2.5 respectively) in the real world case study of [267]. Finally, we note that we set the vehicle capacity Q equal to the value in the original CVRP instance, and we round the transportation costs c up to the nearest integer.

To construct meaningful uncertainty sets Ξ, we assume that each customer $i \in N$ can place a service request on any day of the horizon with probability $\alpha \in [0, 1]$. We then

10 See [193] for references to the original sources of these datasets.

11 We allow for the possibility that $e_i \neq \rho_i$, since the effective day window might have been shrunk in the context of the longer horizon Π (see footnote 5).

consider the following budgeted uncertainty set, which is parameterized by α as well as scalars $\beta, \gamma \in [0,1]$:

$$\Xi = \left\{ \begin{array}{c} \xi^0 = \mathbf{1}, \\ \xi \in \{0,1\}^{|V|} : \sum\limits_{v \in V_p} \xi_v \leq b_p := \alpha n_p + \Phi^{-1}(\gamma)\sqrt{\alpha(1-\alpha)n_p}\ \forall\, p \in P, \\ \sum\limits_{p \in P} \sum\limits_{v \in V_p} \xi_v \leq \beta \sum\limits_{p \in P} b_p \end{array} \right\}. \tag{3.20}$$

In this set, $n_p := \left|V_p\right|$ and Φ is the cumulative distribution function of the standard normal random variable. The budget constraint for period $p \in P$ is based on the central limit theorem (3.4) and stipulates that at most b_p orders will be received in that period with probability γ. In addition, the overall budget imposes that only a fraction β of the periods receive their maximum share of orders. Observe that we recover the deterministic problem by setting $\alpha = 0$ or $\beta = 0$ since in this case, the uncertainty set Ξ becomes a singleton. On the other hand, setting $\beta = 1$ results in an instantiation of a disjoint budget uncertainty set, since the last inequality becomes redundant in this case.

3.6.2 *Computational Performance*

Table 3.1 summarizes the computational performance of the branch-and-cut algorithm of Section 3.4.2 across each of the 95 benchmark instances for various settings of (α, β) and $\Phi^{-1}(\gamma) = 1$ (which corresponds to $\gamma \approx 0.85$). For each setting of (α, β), **Feasible (#)** reports the number of instances (out of 95) which remained feasible in $\overline{\mathcal{TSRO}}$, **Proven optimal (#)** denotes the number of feasible instances for which an optimal solution was found, while **Nodes (#)** and **Time (sec)** respectively report the number of branch-and-bound tree nodes and solution time required, averaged across the same instances. For those feasible instances which could not be solved within two hours, **Gap (%)** reports the average residual gap (defined as $\frac{\text{ub}-\text{lb}}{\text{ub}} \times 100\%$, where ub is the global upper bound and lb is the global lower bound of the branch-and-bound tree). Finally, **No incumbent (#)** reports the number of instances for which the algorithm could neither prove infeasibility nor find a feasible solution in two hours. We make the following three observations from Table 3.1.

1. For a fixed value of β, as the value of α increases, the number of instances which remain feasible in $\overline{\mathcal{TSRO}}$ decreases. This is because the set $\mathcal{S}$ of feasible solutions of model $\overline{\mathcal{TSRO}}$ is monotonous with respect to the uncertainty set Ξ; that is, $\mathcal{S}(\Xi) \subseteq \mathcal{S}(\Xi')$, if $\Xi \supseteq \Xi'$. Therefore, any robust feasible solution under the uncertainty level (α, β) is also feasible under the uncertainty levels (α', β'), where $\alpha' \leq \alpha$ and $\beta' \leq \beta$. Consequently, it is possible that if Ξ becomes too large, the space of robust feasible solutions becomes empty; that is, $\mathcal{S}(\Xi) = \emptyset$.
2. The computational effort involved in solving the robust model is similar to the effort involved in solving the deterministic model. Indeed, if we consider only the

58 instances that remain feasible for all values of (α, β), the deterministic model can be solved for 53 instances using an average time of 380 seconds (the average gap for the unsolved instances being 4.7%). Similarly, the robust model (averaged across all values of $\alpha > 0$) can be solved for 51 instances using an average time of 330 seconds (the average gap for the unsolved instances being 4.4%).

3. In the context of disjoint budget sets, the branch-and-cut method reliably determines a robust feasible solution, if one exists. Indeed, Table 3.1 shows that incumbent solutions are almost always found for the case of $\beta = 1$, as opposed to $\beta = 0.5$, and the corresponding optimality gaps are also smaller. This is in line with Proposition 3.7, which demonstrates that verifying robust feasibility can be done by solving a deterministic bin packing problem, in such cases.

Table 3.1: Summary of computational performance under a time limit of two hours (averaged across all 95 instances for each value of α).

α	Feasible	Proven optimal			Residual gap		No incumbent
	(#)	(#)	Nodes (#)	Time (sec)	(#)	Gap (%)	(#)
				Deterministic ($\beta = 0$)			
0.0	95	68	3,914	447.8	27	5.35	0
				General budget set ($\beta = 0.5$)			
0.2	82	65	3,105	332.1	17	4.31	6
0.4	64	57	3,188	453.5	7	6.33	7
0.6	59	52	4,296	537.9	7	6.08	3
0.8	58	49	5,274	333.5	9	4.73	1
				Cardinality-constrained set ($\beta = 0.5$)			
1.0	58	49	3,170	282.0	9	3.71	1
				Disjoint budget set ($\beta = 1$)			
0.2	73	60	3,363	322.4	13	5.24	0
0.4	64	56	4,350	387.5	8	5.86	0
0.6	60	50	3,751	348.9	10	4.32	0
0.8	58	51	3,458	365.6	7	3.63	1
				Hypercube ($\beta = 1$)			
1.0	58	51	3,904	316.2	7	5.94	0

Tables 3.2 and 3.3 summarize the computational performance as a function of the number of pending orders $|V_0|$. We observe that, across all settings of (α, β) that we considered, all instances with up to 50 pending orders (i.e., $|V_0| \leq 50$) can be solved (to proven optimality or infeasibility) within the imposed time limit. Across instances with $|V_0| > 50$, about 54% of the instances can be solved and the average gap over the remaining unsolved (but feasible) instances is 4.9%. These experiments show that the

two-stage robust model $\overline{\mathcal{TSRO}}$ and the corresponding branch-and-cut method described in Section 3.4.2 are promising for practical applications.

Table 3.2: Summary of computational performance under a time limit of two hours (averaged across all settings of (α, β) for each instance).

$\lvert V_0 \rvert$	(#)	Feasible	Proven optimal			Residual gap		No incumbent
		(#)	(#)	Nodes (#)	Time (sec)	(#)	Gap (%)	(#)
[1, 50]	539	464	464	1,380	84.4	0	–	0
[51, 100]	451	248	142	11,633	1,320.7	106	4.25	15
[101, 150]	33	15	2	1,585	1,289.4	13	9.72	2
[151, 200]	22	2	0	–	–	2	14.78	2
All	1045	729	608	3,775	377.1	121	5.01	19

Table 3.3: Summary of computational performance across a representative set of instances. Optimally solved instances are indicated with an asterisk (along with their computation times); optimality gaps for unsolved instances for which a feasible solution was found are indicated in round brackets; provably infeasible instances are denoted by "Inf"; and, instances for which neither infeasibility could be proved nor a feasible solution could be found are denoted by "NI" (i.e., "No incumbent").

Instance	$\beta = 0$	α $(\beta = 0.5)$					α $(\beta = 1)$				
		0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
E-n51-k5	78.5*	56.6*	56.4*	55.6*	53.6*	53.9*	55.8*	55.6*	53.9*	53.4*	54.0*
E-n76-k10	(2.3%)	(2.6%)	NI	NI	Inf	Inf	(2.9%)	Inf	Inf	Inf	Inf
E-n101-k8	(3.3%)	(2.4%)	(2.8%)	(2.3%)	(5.0%)	(3.5%)	(2.0%)	(3.9%)	(3.3%)	(3.3%)	(3.0%)
M-n151-k12	(7.5%)	NI	NI	Inf	Inf	Inf	(15.1%)	Inf	Inf	Inf	Inf
M-n200-k16	(14.8%)	NI	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf

3.6.3 *Effect of Valid Inequalities*

In order to gain some insight about the effect of various inequalities discussed in Section 3.4.3, Table 3.4 reports six characteristic optimality gaps at the root node (computed with respect to the final incumbent solution), as follows: (G0) the gap obtained using the initial linear relaxation, as described in the preamble of Section 3.4.2; (G1) the gap obtained after separation of the rounded capacity inequalities (RCI) (3.10h); (G2) the gap obtained after separation of the RCI and robust cover inequalities (3.11), (3.11′); (G3) the gap obtained after separation of the RCI, robust cover and cumulative capacity

inequalities (3.13); (G4) the gap obtained after separation of the RCI, robust cover, cumulative capacity and CVRP inequalities (see Proposition 3.6); and finally, (G5) the gap obtained after separation of all inequalities, including the generalized routing-related inequalities (3.14)–(3.15).

Table 3.4: Root node gaps (%) obtained after various levels of separation of valid inequalities (averaged across all settings of α and all instances for which a feasible solution was found).

β		Necessary cuts		Strengthening cuts		
	G0	G1	G2	G3	G4	G5
0.0	40.2	12.7	12.7	12.7	12.3	5.3
0.5	38.7	12.5	12.5	12.5	12.2	5.1
1.0	38.5	12.7	12.7	12.6	12.3	5.4

We observe from Table 3.4 that the initial relaxations are very weak (gaps G0 $\approx$ 40%), but they improve significantly, to roughly 13%, when the *necessary* RCI and robust cover inequalities are added as cutting planes. The addition of the *strengthening* inequalities further reduces the gaps to about 5%, on average. Evidently, the routing-related RCI, other CVRP-based inequalities and their generalized versions are very effective and important at the root node. We note, however, that the robust cover and cumulative capacity inequalities are also effective, albeit at deeper nodes of the search tree. To see this, Table 3.5 reports the number of different families of inequalities identified during the entire branch-and-cut algorithm. In order to obtain meaningful statistics, we only average across the "hard" instances; that is, those which could not be solved within 300 seconds. We observe from this table that the fraction of robust cover and cumulative capacity inequalities as a percentage of the total number of cuts, is comparable to the fraction of routing-related inequalities, indicating that they are essential for the computational efficiency of the branch-and-cut algorithm.

Table 3.5: Time spent in separation algorithms and number of different families of cuts added (averaged across all settings of α and all instances with solution time $>$ 300 seconds).

β	Sep time	Cuts	Necessary cuts (%)		Strengthening cuts (%)		
	(%)	(#)	RCI	Robust cover	Cum capacity	CVRP	GSEC/GFCI
0.0	11.1	19,119	69.3	0.0	0.0	7.2	23.5
0.5	34.8	16,360	50.0	11.5	16.9	5.5	16.1
1.0	15.0	18,539	60.4	10.0	6.5	5.0	18.1

Table 3.5 also reports the time spent in separation algorithms as a percentage of the total computation time (see Section 3.4.4). It is evident that separation time is a major component of the total computing time in the robust setting, especially in the case of

uncertainty sets for which Proposition 3.7 is not applicable (as is the case for $\beta = 0.5$), since in such cases, we must resort to a column-and-constraint generation algorithm to verify robust feasibility of candidate solutions.

Finally, Table 3.6 summarizes the overall performance of the branch-and-cut algorithm with and without the valid inequalities described in Section 3.4.3. The different rows represent increasingly aggressive levels of cut separation in the algorithm. At the bare minimum, when no cuts except the necessary RCI and robust cover inequalities are separated at only integral nodes of the branch-and-bound tree, only 289 out of 729 instances can be solved to optimality and the average gap for the remaining 440 instances is 27%. In contrast, 608 out of 729 instances can be solved when all possible inequalities are separated at all tree nodes, with the remaining unsolved instances proved to be within 5% of their optimal values.

Table 3.6: Summary of computational performance with and without the valid inequalities described in Section 3.4.3 (averaged across all 729 instances from Table 3.2 for which a feasible solution was found).

Separation intensity	Proven optimal			Residual gap	
	(#)	Nodes (#)	Time (sec)	(#)	Gap (%)
Only necessary cuts at only integral nodes	289	240,540	550.0	440	27.31
Only necessary cuts at all nodes	540	9,124	694.6	189	9.26
Necessary and strengthening cuts at all nodes	608	3,775	377.1	121	5.01

3.6.4 *Approximation Quality and Price of Robustness*

Since the two-stage robust model $\overline{\mathcal{TSRO}}$ is a conservative approximation to the fully adaptive multistage robust model $\mathcal{MSRO}$, the corresponding solution thus obtained is feasible–but may not be optimal–for the fully adaptive model. In this section, we aim to estimate the approximation quality of the solutions provided by the two-stage robust model $\overline{\mathcal{TSRO}}$. Proposition 3.1 shows that this quantity can be bounded from above as follows: $\left(\overline{\texttt{TSRO}} - \underline{\texttt{TSRO}}\right)/\underline{\texttt{TSRO}}$, where $\underline{\texttt{TSRO}}$ is the optimal objective value of the two-stage robust model $\underline{\mathcal{TSRO}}$. Section 3.5 shows how to obtain $\underline{\texttt{TSRO}}$ through a branch-and-cut method similar to the one described in Section 3.4.2. Since obtaining the globally optimal value $\underline{\texttt{TSRO}}$ through this branch-and-cut method may be computationally challenging, we employ a large time limit of 12 hours and estimate $\underline{\texttt{TSRO}}$ using the global lower bound of the branch-and-bound tree at termination, thus ensuring that the corresponding estimate is always lower that the optimal value of the fully adaptive multi-stage model. Similarly, we estimate $\overline{\texttt{TSRO}}$ using the global upper bound of its branch-and-bound tree at termination (i.e., using the objective value of the incumbent from Section 3.6.2). This

guarantees that the reported values of the *approximation gap* in Table 3.7 are conservative estimates of the true values.

Table 3.7 also reports average values of the *price of robustness*, which is the increase in cost of the optimal solution of the two-stage robust model $\overline{\mathcal{TSRO}}$ over that of the deterministic model under the nominal realization, $\mathcal{DET}(\hat{\xi})$, and is defined as follows: $\left(\underline{\texttt{TSRO}} - \texttt{DET}(\hat{\xi})\right)/\texttt{DET}(\hat{\xi})$. As before, if the optimal objective value of the deterministic model is not available, we estimate it with the corresponding global lower bound of the branch-and-bound tree at termination, thus ensuring that the reported values of the price of robustness are upper bounds to the true values.

Table 3.7: Price of robustness and guaranteed approximation gap under different settings of (α, β), averaged across the 58 instances from Table 3.1 which remain feasible under the highest setting of $(\alpha, \beta) = (1, 1)$.

α	0.2	0.4	0.6	0.8	1.0
	General budget set ($\beta = 0.5$)				
Price of robustness	0.32	0.40	0.62	0.96	1.05
Approximation gap	0.28	0.37	0.58	0.92	1.01
	Disjoint budget set ($\beta = 1$)				
Price of robustness	0.32	0.53	0.93	0.96	1.31
Approximation gap	0.28	0.50	0.89	0.82	0.92

We make the following observations from Table 3.7. First, the average price of robustness varies between 0.3% to 1.3%, implying that, for the considered benchmark instances, solutions which are robust against vehicle capacity violations can be obtained at marginal cost increases above the deterministic solutions. Second, the two-stage model $\overline{\mathcal{TSRO}}$ provides a good approximation of the multi-stage fully adaptive model $\mathcal{MSRO}$, since the average approximation gap is consistently less than 1%, on average. Furthermore, across instances for which $\overline{\mathcal{TSRO}}$ was infeasible (not shown in Table 3.7), the two-stage model $\underline{\mathcal{TSRO}}$ was also infeasible for about 53% and 85% of the instances, under $\beta = 0.5$ and $\beta = 1$ respectively, implying that the corresponding multi-stage model $\mathcal{MSRO}$ was also infeasible for these instances. This supports the claim in Proposition 3.3 that the two-stage model $\overline{\mathcal{TSRO}}$ is expected to provide a good approximation of the multi-stage model, and especially so in the case of disjoint budget uncertainty sets, where $\beta = 1$.

3.6.5 *Comparison with Existing Methods using Rolling Horizon Simulations*

In this section, we aim to study the performance of the various models in real-time operations. To achieve this goal, we develop a rolling horizon simulation platform to mimic the daily operations faced by distributors, where information about uncertain customer orders is revealed sequentially over time. We conduct Monte-Carlo simulations

of four different models over a 30-day horizon, that is, $|\Pi| = 30$. At the end of each day $p \in \Pi$ of the simulation, the selected model is solved by considering the set of all orders pending up to (and received on) day p, to determine the set of orders as well as the routes that will be executed on day $p+1$. Then, the actual transportation cost incurred on day $p+1$ is recorded, the time horizon rolls forward, and the same procedure is repeated at the end of day $p+1$. For each simulation, we consider the following decision approaches:

(i) *Early policy*, which always serves each pending order on the first day of its day window,

(ii) *Late policy*, which always serves each pending order on the last day of its day window,[12]

(iii) *Deterministic model* $\mathcal{DET}(\hat{\xi})$ with a five-day planning horizon ($h = 5$), and

(iv) *Robust model* $\overline{\mathcal{TSRO}}$ with a five-day planning horizon ($h = 5$) and budgeted uncertainty set.

In the case of the deterministic and robust models, the planning horizon shrinks in the last five days of the simulation. We note that the early and late policies are very popular in industry because of their simplicity, and they resemble current industrial practices. Hence, it is natural to benchmark our proposed method against these approaches.

Each of the above decision-making models ignores information beyond its current planning horizon. Therefore, it is possible that the selected model can become infeasible on some day of the simulation, as the fleet size and vehicle capacities are limited. To monetize infeasibility in such cases, we consider a per-unit penalty cost M of commissioning additional vehicles and solve the following penalized model to recover implementable routes:

$$\begin{aligned} \underset{\mathbf{S},\theta}{\text{minimize}} \quad & \sum_{p\in P} \texttt{CVRP}(S_p, \hat{\xi}, m+\theta_p) + M \sum_{p\in P} \theta_p \\ \text{subject to} \quad & (S_1,\ldots,S_h) \in \mathcal{F}, \;\; \theta \in \mathbb{Z}_+^p \\ & \texttt{BPP}(S_p,\xi) \le m+\theta_p \quad \forall\, p \in P,\, \forall\, \xi \in \Xi, \end{aligned}$$

Here, θ_p is a slack variable representing the number of additional vehicles required on day $p \in P$. In practice, M is typically very large and hence, we can equivalently solve the penalized model by lexicographically minimizing the number of additional vehicles first, and then minimizing the vehicle routing cost. Specifically, in the first phase, we determine the minimum total number of additional vehicles that need to be commissioned, $\Theta^\star$, by solving the above model without the routing-related CVRP terms in the objective function. In the second phase, we solve the same model by setting $M = 0$, but with an additional constraint on the total number of additional vehicles that can be used, $\sum_{p\in P} \theta_p \le \Theta^\star$. In the simulation, the final cost recorded is the sum of the routing

12 We note that this is equivalent to the deterministic model $\mathcal{DET}(\hat{\xi})$ with a planning horizon of one day ($h = 1$).

cost on day 1, $\texttt{CVRP}(S_1, \hat{\xi}, m + \theta_1)$, and the incurred penalty cost $M\theta_1$ associated with commissioning θ_1 additional vehicles. We note that, in the above optimization model, $\mathcal{F}$ and Ξ are singletons and $P = \{1\}$ for the early and late policies, whereas Ξ is a singleton for the deterministic model.

For our simulations, we randomly select 10 instances from Section 3.6.1 consisting of up to 50 customers, $|N| \leq 50$. For each instance, customer orders are simulated as independent Bernoulli processes with probabilities $\in \{0.25, 0.30, 0.35\}$, which represent low, medium and high levels of customer demand. For each combination of test instance, decision-making model and probability level, we carry out 100 rounds of simulations. We emphasize that the same realization of customer orders are used in each round of the simulation to ensure a fair comparison between the different models. For the robust model $\overline{\mathcal{TSRO}}$, we select the uncertainty set (3.20) with $\beta = 0$ and various levels of $\gamma \in \{50\%, 85\%, 90\%, 95\%\}$. The value of α is chosen to be the same as the corresponding probability level, although this can be easily replaced by a quantity statistically estimated from (and updated using) data, whenever it is available. We set $M = 100\bar{c}$, where $\bar{c}$ is the maximum value of the travel costs c_{ij} across all $(i, j) \in E$, and a time limit of two hours per optimization run.

Table 3.8 summarizes the simulation performance of the early, late, deterministic ("Det") and robust approaches for various ordering levels α. For each approach, each setting of α and each of the 10 test instances, we compute the average total cost, worst-case total cost, average routing cost and average penalty cost incurred over the simulation horizon (the average and worst-case being taken across the 100 simulation rounds), and then normalize each of them with respect to average total cost incurred by the deterministic model. The average (across the 10 test instances) of these normalized quantities are then reported in columns **Total cost avg**, **Total cost wc**, **Routing cost avg** and **Penalty cost avg**, respectively. The table also reports for each approach and each setting of α, the percentage of simulations in which penalties were incurred, averaged across the 10 instances, in column **Penalty freq avg (%)**. We make the following observations from Table 3.8.

1. The early and late policies are strongly outperformed by the deterministic and robust models. Compared to the latter, they incur about 50% and 220% more cost under the lowest and highest ordering levels, respectively. This is because they are "myopic": they ignore not only uncertain future information, but also existing knowledge of the pending orders (except their first or last feasible service days). In contrast, the deterministic and robust models "look ahead" to serve geographically close orders with overlapping day windows in the same route.
2. The robust model at $\gamma = 50\%$ achieves the best overall performance. The outperformance with respect to the deterministic model is particularly pronounced at higher ordering levels. Indeed, at $\alpha = 0.35$, it incurs a total cost which is about 5% lower on average and 26% lower in the worst-case, and it experiences a 49% lower

Table 3.8: Summary of simulation performance (averaged across 100 rounds for each of 10 test instances).

				Robust (γ)			
Decision approach	Early	Late	Det	50%	85%	90%	95%
			$\alpha = 0.25$				
Total cost avg	151.6	148.9	100.0	99.7	99.9	99.9	100.1
Total cost wc	370.0	369.3	126.7	111.4	111.8	111.9	111.8
Routing cost avg	119.6	116.5	99.7	99.7	99.9	99.9	100.1
Penalty cost avg	31.9	32.4	0.3	0.0	0.0	0.0	0.0
Penalty freq avg (%)	16.1	16.3	0.2	0.0	0.0	0.0	0.0
			$\alpha = 0.30$				
Total cost avg	226.1	225.5	100.0	98.4	99.0	99.2	99.5
Total cost wc	558.3	581.6	177.9	123.6	135.3	136.5	147.3
Routing cost avg	116.7	113.6	97.8	98.1	98.5	98.6	98.6
Penalty cost avg	109.5	111.9	2.2	0.3	0.5	0.6	0.9
Penalty freq avg (%)	43.8	44.5	1.5	0.2	0.4	0.4	0.7
			$\alpha = 0.35$				
Total cost avg	336.2	326.0	100.0	95.4	97.0	97.1	97.2
Total cost wc	741.9	702.2	214.1	158.8	170.5	169.2	182.7
Routing cost avg	106.7	103.8	89.8	90.5	90.7	90.6	90.6
Penalty cost avg	229.5	222.2	10.2	4.9	6.3	6.5	6.6
Penalty freq avg (%)	63.4	63.5	9.0	4.6	5.7	5.9	5.9

Note. Costs are normalized with respect to the "Total cost avg" entry of the "Det" column, for each α.

frequency of infeasibility. Moreover, these savings come at the price of incurring only 0.9% higher routing costs, on average.

3. Choosing a higher value of γ does not necessarily translate to lower frequencies of infeasibility. While it is true that a larger uncertainty set leads to a higher probability of being feasible in a folding horizon context (all else being equal), this is not true in a rolling horizon context. This is because *(i)* the models ignore information beyond the current planning horizon, which is revealed only when time rolls forward, and *(ii)* on a particular day $p > 1$, the models do not necessarily solve the same problem, since their previous decisions lead to different instantiations of the problem parameters (e.g., the set of pending orders). A similar discussion can be found in [135] in the context of general robust optimization problems.

3.7 SUMMARY

In this chapter, we studied the robust multi-period VRP under customer order uncertainty, in which customers call-in to request service over a short-term planning horizon, specifying a set of days during which the service can take place. The decision-maker aims to select a visit schedule over the planning horizon that remains feasible for all realizations of uncertain customer orders, which were modeled as binary random variables supported on a finite uncertainty set. The true multi-stage decision-making process was approximated from above and below by two two-stage robust optimization models. For each approximation, an integer programming formulation was derived and a numerically efficient branch-and-cut solution technique was presented. Extensive computational experiments conducted on a number of test instances derived from standard benchmark datasets showed that robust feasible solutions can be obtained with a computational effort similar to that for nominal solutions, and that are of high quality, on average. A rolling horizon simulation study further showed that the robust solutions outperform the nominal solutions in terms of reducing the frequency of vehicle capacity violations while incurring only marginally higher routing costs.

3.8 APPENDIX: NOMENCLATURE

Π	Time horizon
m	Number of vehicles available on each day $d \in \Pi$
K	Set of vehicles available on each day $d \in \Pi$
Q	Capacity of each vehicle
N	Set of customers
N'	Set of customers and the depot $N \cup \{0\}$
$G = (N', E)$	Undirected graph with node set N' and edge set E
$v = (i, d)$	Order placed by customer $i \in N$ on day $d \in \Pi$
P_v	Day window of order $v = (i, d)$, given by $\{d + e_i, \ldots, d + \ell_i\}$
q_i, q_v	Demand of customer $i \in N$ or any order $v = (i, \cdot)$
c_{ij}, c_{uv}	Routing cost along $(i, j) \in E$ or between $u = (i, \cdot)$ and $v = (j, \cdot)$
h	Length of planning horizon
P	Set of days in planning horizon
V_0	Set of pending orders at the end of day $0 \in \Pi$
E_0	Set of edges in E covering customers with a pending order
$E_0(S)$	Subset of edges in E_0 covering orders in $S \subseteq V_0$
V_p	Set of potential orders that can be received on day $d \in \Pi$
V	Set of pending and potential future orders $= V_0 \cup V_1 \cup \ldots \cup V_h$
$\mathcal{F}$	Set of all assignments of customer orders in V to periods in P
Δ_p	Set of orders in V that can be visited on day $p \in P$
S_p	Subset of orders in Δ_p selected to be served on day $p \in P$
ξ_v, ξ_{ip}	Binary random variable associated with order $v = (i, p)$
$\hat{\xi}$	Nominal realization defined as $\hat{\xi}_v = 1$ if $v \in V_0$ and 0 otherwise
Ξ	Uncertainty set of customer orders
L, B, b	Parameters used to define budget uncertainty sets Ξ_B
y_{vp}	Binary variable $\in \{0, 1\}$ indicating if order $v \in V$ is served on day $p \in P$
x_{uvp}	Binary variable $\in \{0, 1\}$ indicating if order $u \in V_0$ is served after order $v \in V_0$ on day $p \in P$
x_{0vp}	Integer variable $\in \{0, 1, 2\}$ recording the number of times edge $(0, i) \in E_0$ is traversed on day $p \in P$, where $v = (i, \cdot)$
θ_p	Integer variable recording the number of additional vehicles required on day $p \in P$
α, β, γ	Parameters used to define the uncertainty sets in Section 3.6

$\texttt{CVRP}(S, \xi, m)$	Optimal value of a capacitated vehicle routing problem (defined in Section 4.2)
$\texttt{BPP}(S, \xi)$	Optimal value of a bin packing problem (defined in Section 4.2)
$\texttt{BPPDW}(S, \xi)$	Optimal value of a bin packing problem with day windows (defined in Section 3.5)
$\mathcal{DET}(\hat{\xi})$	Deterministic model
$\mathcal{MSRO}$	Multi-stage robust optimization model
$\overline{\mathcal{TSRO}}$	Non-anticipative two-stage robust optimization model
$\overline{\mathcal{TSRO}}_{IP}$	Integer programming formulation of $\overline{\mathcal{TSRO}}$
$\underline{\mathcal{TSRO}}$	Anticipative (lower bounding) two-stage robust optimization model

3.9 APPENDIX: PROOFS OF PROPOSITIONS

Proof of Proposition 3.1. The first inequality follows from the observation that $\underline{\mathcal{TSRO}}$ is equivalent to the following problem:

$$
\begin{aligned}
\underset{\mathbf{S},\tilde{\mathbf{R}}}{\text{minimize}} \quad & \sum_{p\in P} \texttt{CVRP}(S_p, \hat{\xi}, m) \\
\text{subject to} \quad & (S_1, S_2, \ldots, S_h) \in \mathcal{F} \\
& \texttt{BPP}(S_p, \hat{\xi}) \leq m \quad \forall\, p \in P \setminus \{1\} \\
& \tilde{R}_p : \Xi \setminus \{\hat{\xi}\} \mapsto 2^{\Delta_p} \quad \forall\, p \in P \setminus \{1\} \\
& (S_1, \tilde{R}_2(\xi), \ldots, \tilde{R}_h(\xi)) \in \mathcal{F} \quad \forall\, \xi \in \Xi \setminus \{\hat{\xi}\} \\
& \texttt{BPP}(\tilde{R}_p(\xi), \xi) \leq m \quad \forall\, p \in P \setminus \{1\},\ \forall\, \xi \in \Xi \setminus \{\hat{\xi}\}
\end{aligned}
$$

This description differs from the original one in that it has explicit decision variables for the assignments resulting from evaluating the functionals $\tilde{S}_p(\cdot)$ under the nominal scenario. Observe that the second constraint in the above problem is redundant since it is embedded in the evaluation of $\texttt{CVRP}(S_p, \hat{\xi}, m)$ in the objective function. Therefore, $\mathcal{DET}(\hat{\xi})$ is obtained by relaxing the last three constraints and its optimal value provides a lower bound to that of $\underline{\mathcal{TSRO}}$.

The second inequality follows from the fact that for any feasible solution $(S_1, \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ in $\mathcal{MSRO}$, we can construct a feasible solution $(S'_1, S'_2(\cdot), \ldots, S'_h(\cdot))$ in $\underline{\mathcal{TSRO}}$ as follows: $S'_1 = S_1$ and $S'_p(\xi) \triangleq \tilde{S}_p(\xi^{[p-1]})$ for all $\xi \in \Xi$. Therefore, the feasible region of $\mathcal{MSRO}$ is a subset of the feasible region of $\underline{\mathcal{TSRO}}$ and its optimal value provides an upper bound to that of $\underline{\mathcal{TSRO}}$.

The third inequality follows if, for all $p > 1$, we restrict the functional variables $\tilde{S}_p(\cdot)$ in $\mathcal{MSRO}$ to the space of constant functions. In doing so, we obtain $\overline{\mathcal{TSRO}}$, whose optimal value provides an upper bound to the optimal value of $\mathcal{MSRO}$. □

Proof of Proposition 3.2. In either case, the optimal functions $\tilde{S}_p(\cdot)$, for all $p \in P \setminus \{1\}$, in both models $\underline{\mathcal{TSRO}}$ and $\mathcal{MSRO}$ can be obtained as follows: $\tilde{S}_p(\cdot) \equiv \{v \in V : p \in P_v, v \notin S_1\}$, i.e., they are constant over their domain. Therefore, the functional variables $\tilde{S}_p : \Xi \mapsto 2^{\Delta_p}$ in $\underline{\mathcal{TSRO}}$ and $\tilde{S}_p : \Xi^{[p-1]} \mapsto 2^{\Delta_p}$ in $\mathcal{MSRO}$ can be replaced with the decision variables $S_p \in 2^{\Delta_p}$ (i.e., $S_p \subseteq \Delta_p$) without loss of optimality. In both cases, the model obtained by restricting the functional variables to the space of constant functions is optimal and, therefore, equivalent to $\overline{\mathcal{TSRO}}$. □

Proof of Proposition 3.3. The stated result clearly holds under conditions of Proposition 3.2 for which $\texttt{MSRO} = \overline{\texttt{TSRO}}$.

Consider now case *(i)*. Assume that $(S_1, \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ is a feasible solution in model $\mathcal{MSRO}$. Consider the order assignment $\mathbf{S} = (S_1, \tilde{S}_2(\mathbf{1}^{[1]}), \ldots, \tilde{S}_h(\mathbf{1}^{[h-1]}))$ obtained by evaluating this feasible solution under the worst-case uncertainty realization,

$\mathbf{1} \in \Xi$, where Ξ is a hypercube. This assignment is also feasible in model $\overline{\mathcal{TSRO}}$ since *(a)* $\mathbf{S} \in \mathcal{F}$, as per the second constraint in $\mathcal{MSRO}$, and *(b)* $\texttt{BPP}(\tilde{S}_p(\mathbf{1}^{[p-1]}), \xi) \leq \texttt{BPP}(\tilde{S}_p(\mathbf{1}^{[p-1]}), \mathbf{1}) \leq m$ is satisfied for all $\xi \in \Xi$; indeed, the second inequality follows from the third constraint in model $\mathcal{MSRO}$, while the first inequality follows from the fact that the optimal value of the bin-packing problem associated with any set is always no larger than the optimal value associated with any subset.

Consider now case *(ii)*. Define $\underline{\xi} = (\xi^0, \underline{\xi}^1, \ldots, \underline{\xi}^{h-1}) \in \Xi$ as follows: $\underline{\xi}^p := (\xi_v[p])_{v \in V_p}$, where $\xi[p]$ is an optimal solution of $\max_{\xi \in \Xi} \texttt{BPP}(S_p, \xi)$, where $S_p := \{v \in V : P_v = \{p\}\}$ for all $p \in P \setminus \{1\}$. Similar to case *(ii)*, assume that $(R_1, \tilde{R}_2(\cdot), \ldots, \tilde{R}_h(\cdot))$ is a feasible solution in model $\mathcal{MSRO}$. Now, consider the $\mathbf{R} = (R_1, \tilde{R}_2(\underline{\xi}^{[1]}), \ldots, \tilde{R}_h(\underline{\xi}^{[h-1]}))$ to be the order assignment obtained by evaluating this feasible solution under the uncertainty realization $\underline{\xi} \in \Xi$. This assignment is also feasible in model $\overline{\mathcal{TSRO}}$ since *(a)* $\mathbf{R} \in \mathcal{F}$, as per the second constraint in $\mathcal{MSRO}$, and *(b)* $\texttt{BPP}(\tilde{R}_p(\underline{\xi}^{[p-1]}), \xi) \leq \texttt{BPP}(\tilde{R}_p(\underline{\xi}^{[p-1]}), \underline{\xi}) \leq m$ for all $\xi \in \Xi$; indeed, the second inequality follows from the third constraint in $\mathcal{MSRO}$, while the first inequality follows from the fact that $\underline{\xi}$ is an optimal solution of $\max_{\xi \in \Xi} \texttt{BPP}(\tilde{R}_p(\underline{\xi}^{[p-1]}), \xi)$. We remark that the latter holds because the proof of Proposition 3.7 shows that this optimal value only depends on orders in $\tilde{R}_p(\underline{\xi}^{[p-1]}) \bigcap_{q \in P, l} C_{lq} = S_p$. In turn, this last equality holds because all customers $i \in N$ with $w_i \geq 2$ satisfy $(i, q) \notin C_{lq}$ for any l, q (since all orders from i are preprocessed from the uncertainty set, i.e., $(i, q) \notin V$ for any $q \in P$), and any customer order $v = (i, q)$ with $w_i = 1$ and $q \neq p - 1$ satisfies $v \notin \tilde{R}_p(\underline{\xi}^{[p-1]})$ (since $e_i = 1$ for all such orders). □

Proof of Proposition 3.4. Necessity. Let $\mathbf{S}(y)$ be any robust feasible solution in $\overline{\mathcal{TSRO}}$ that is induced by binary variables y. Also, let $S \subseteq V$, $p \in P$ and $\xi \in \Xi$ be given. Then we have that:

$$
\begin{aligned}
m + \sum_{i \in S} (1 - y_{ip}) &= m + \sum_{i \in S \cap S_p(y)} (1 - 1) + \sum_{i \in S \setminus S_p(y)} (1 - 0) \\
&= m + \left| S \setminus S_p(y) \right| \\
&\geq \texttt{BPP}(S_p(y), \xi) + \left| S \setminus S_p(y) \right| \\
&\geq \texttt{BPP}(S \cap S_p(y), \xi) + \left| S \setminus S_p(y) \right| \\
&\geq \texttt{BPP}(S \cap S_p(y), \xi) + \texttt{BPP}(S \setminus S_p(y), \xi) \\
&\geq \texttt{BPP}(S, \xi)
\end{aligned}
$$

Here, the first equality follows from (3.8). The first inequality follows from the definition of robust feasibility of $\mathbf{S}(y)$ in $\overline{\mathcal{TSRO}}$. The second inequality follows from the fact that the optimal value of the bin packing problem over a given set of items is always no smaller than its optimal value over any subset of items. The third inequality follows from the fact that the cardinality of the set of items is a trivial upper bound to the optimal value of the corresponding bin packing problem. Finally, the last inequality follows from the fact that the optimal value of the bin packing problem possesses the subadditivity property.

Sufficiency. Assume that y satisfies equations (3.9) and the robust cover inequalities (3.11). We must then show that $\mathbf{S}(y)$ is a robust feasible solution in $\overline{\mathcal{TSRO}}$. From equations (3.8)–(3.9), we know that $\mathbf{S}(y)$ partitions V; that is, $\mathbf{S}(y) \in \mathcal{F}$. We therefore only need to verify that, for every $\xi \in \Xi$ and $p \in P$, the relationship $\texttt{BPP}(S_p(y), \xi) \leq m$ holds. This is true since the left-hand side of (3.11) for $S = S_p(y)$ evaluates to m, whereas the right-hand side evaluates to $\texttt{BPP}(S_p(y), \xi)$. □

Proof of Proposition 3.5. Instead of proving the proposition directly, we prove its contraposition: if inequality (3.12) associated with $p \in P$ is violated by some solution (x^*, y^*) of the constraint system (3.10b)–(3.10h), then at least one of the robust cover inequalities (3.11) associated with p is also violated by (x^*, y^*). Therefore, let us assume that $\sum_{i \in S} \tilde{\xi}_i (1 - y^*_{ip}) - \sum_{i \in N(S)} \tilde{\xi}_i y^*_{ip} < k$ is satisfied.

Consider the robust cover inequality (3.11) defined by period p, customer order realization $\tilde{\xi}$, and the subset $S' = S^+ \cup NS^+$, where $S^+ = \{i \in S \cap S_p(y^*) : \tilde{\xi}_i = 1\}$ and $NS^+ = \{i \in N(S) \cap S_p(y^*) : \tilde{\xi}_i = 1\}$. We shall prove that this inequality is violated by (x^*, y^*). Observe that the left-hand side of this inequality evaluates to m since $i \in S_p(y^*)$ is satisfied for all $i \in S'$ by construction. Therefore, we would like to show that its right-hand side exceeds m; that is, $\texttt{BPP}(S', \tilde{\xi}) \geq m + 1$.

We define the set $S^1 = \{i \in S : \tilde{\xi}_i = 1\}$ for ease of notation. Note that $\texttt{BPP}(S^1, \tilde{\xi}) = \texttt{BPP}(S, \tilde{\xi}) \geq m + k$, where $k \geq 1$, holds by hypothesis. We now consider two different cases:

(i) $|S^+| > |S^1| - k$.
Observe that, since $S^+ \subseteq S^1$ and $|S^+| > |S^1| - k$, S^+ is obtained from S^1 after the removal of fewer than k elements. Therefore, the optimal bin packing value associated with S^+ can only decrease by some number less than k with respect to the optimal bin packing value associated with S^1; that is, $\texttt{BPP}(S^+, \tilde{\xi}) \geq \texttt{BPP}(S^1, \tilde{\xi}) - (k-1) \geq m + k - (k-1) = m + 1$. Since $\texttt{BPP}(S', \tilde{\xi}) \geq \texttt{BPP}(S^+, \tilde{\xi})$, we have that $\texttt{BPP}(S', \tilde{\xi}) \geq m + 1$.

(ii) $|S^+| \leq |S^1| - k$.
Let S^- be any non-empty subset of $S^1 \setminus S^+$ such that $|S^-| = |S^1| - |S^+| - (k-1)$. It is possible to construct such a subset since $|S^+| \leq |S^1| - k$ and $k \geq 1$. The following holds:

$$\begin{aligned}
\texttt{BPP}(S', \tilde{\xi}) &= \texttt{BPP}(S^+ \cup NS^+, \tilde{\xi}) \\
&\geq \texttt{BPP}(S^+ \cup S^-, \tilde{\xi}) \\
&\geq \texttt{BPP}(S^1, \tilde{\xi}) - (k-1) \\
&\geq m + 1
\end{aligned}$$

The first inequality follows from the following two observations: *(i)* our hypothesis is $\sum_{i \in S} \tilde{\xi}_i (1 - y^*_{ip}) - \sum_{i \in N(S)} \tilde{\xi}_i y^*_{ip} < k$; this is equivalent to $|NS^+| > |S^1| - |S^+| - k = |S^-|$, and *(ii)* each item of NS^+ has higher weight (i.e., demand) than each

item of S^- since $NS^+ \subseteq N(S)$. The second inequality follows from the fact that $S^+ \cup S^- \subseteq S^1$ and $S^+ \cup S^-$ is constructed from S^1 by the removal of $k-1$ customers. Therefore, the optimal bin packing value associated with $S^+ \cup S^-$ can only decrease by at most $k-1$ with respect to the optimal value associated with S^1. The last inequality follows from $\texttt{BPP}(S^1, \tilde{\xi}) \geq m + k$, where $k \geq 1$.

□

Proof of Proposition 3.6. Consider the following relaxation of formulation (3.10) obtained by relaxing the day window constraints (3.10e) and the fleet availability constraints (3.10f):

$$
\begin{aligned}
(3.10)' \quad \underset{x,y}{\text{minimize}} \quad & \sum_{p \in P} \sum_{v \in V_0} c_{0v} x_{0vp} + \sum_{p \in P} \sum_{(u,v) \in E_0} c_{uv} x_{uvp} & \\
\text{subject to} \quad & y_{vp} \in \{0,1\} & \forall\, v \in V,\ \forall\, p \in P \\
& x_{0vp} \in \{0,1,2\} & \forall\, v \in V_0,\ \forall\, p \in P \\
& x_{uvp} \in \{0,1\} & \forall\, (u,v) \in E_0,\ \forall\, p \in P \\
& \sum_{p \in P} y_{vp} = 1 & \forall\, v \in V_0 \\
& x_{0vp} + \sum_{u:(u,v) \in E_0} x_{uvp} = 2 y_{vp} & \forall\, v \in V_0,\ \forall\, p \in P \\
& \sum_{p \in P} \sum_{(u,v) \in E_0(S)} x_{uvp} \leq |S| - \left\lceil \frac{1}{Q} \sum_{i \in S} q_i \right\rceil & \forall\, S \subseteq V_0
\end{aligned}
$$

Now, if we add the two-index variables to the above formulation, via the identities $x'_{uv} = \sum_{p \in P} x_{uvp}$ for all $(u,v) \in E_0 \cup \{(0,v') : v' \in V_0\}$, and project the resulting integer polytope into the space of the two-index variables x', we obtain the integer polytope associated with the following formulation:

$$
\begin{aligned}
\underset{x'}{\text{minimize}} \quad & \sum_{v \in V_0} c_{0v} x'_{0v} + \sum_{(u,v) \in E_0} c_{uv} x'_{uv} & \\
\text{subject to} \quad & x'_{0v} \in \{0,1,2\} & \forall\, v \in V_0 \\
& x'_{uv} \in \{0,1\} & \forall\, (u,v) \in E_0 \\
& x_{0v} + \sum_{u:(u,v) \in E_0} x_{uv} = 2 & \forall\, v \in V_0 \\
& \sum_{(u,v) \in E_0(S)} x_{uv} \leq |S| - \left\lceil \frac{1}{Q} \sum_{i \in S} q_i \right\rceil & \forall\, S \subseteq V_0
\end{aligned}
$$

This is precisely the two-index formulation of the CVRP instance defined on the subgraph of G with depot node 0, customers V_0, edges E_0, demands q_v for $v \in V_0$ and vehicle capacity Q. Therefore, if $\sum_{(i,j) \in I} \lambda_{ij} x'_{ij} \leq \mu$ is any valid inequality for the above two-index formulation, the inequality $\sum_{p \in P} \sum_{(i,j) \in I} \lambda_{ij} x_{ijp} \leq \mu$ is valid for formulation (3.10)′ and, hence, valid for formulation (3.10), since the former constitutes a relaxation of the latter. □

Proof of Proposition 3.7. We consider values of ξ_v^*, $v \in V$, for each of the following cases separately:

1. $v \notin (V_0 \cup S)$.
   Observe that, for given $\xi \in \Xi$, we can set $\xi_v = 0$ without changing the value of $\texttt{BPP}(S, \xi)$ and without affecting the validity of $\xi \in \Xi$ (since each budget constraint continues to be satisfied). Therefore, there always exists an optimal solution that satisfies $\xi_v = 0$ for all $v \notin (V_0 \cup S)$.

2. $v \in (V_0 \cup S) \setminus \left(\cup_{l=1}^{L} B_l\right)$. This is equivalent to $v \in J_0$, by construction.
   Observe that, if $\xi_v = 0$, then setting $\xi_v = 1$ does not decrease the value of $\texttt{BPP}(S, \xi)$ and, moreover, we do not violate $\xi \in \Xi$ (since ξ_v does not appear in any budget constraint). Therefore, any maximizer ξ^* of $\texttt{BPP}(S, \xi)$ must also satisfy $\xi_v^* = 1$ for all $v \in J_0$.

3. $v \in (V_0 \cup S) \cap \left(\cup_{l=1}^{L} B_l\right)$. This is equivalent to $v \in S \cap \left(\cup_{l=1}^{L} B_l\right)$ since, by construction, we have $V_0 \cap \left(\cup_{l=1}^{L} B_l\right) = \emptyset$.
   Since the subsets $\{B_l\}_{l=1}^{L}$ are disjoint by assumption, assume that $v \in S \cap B_l$, where $1 \leq l \leq L$ is some budget, and that $v \notin B_{l'}$, for all $l' \neq l$. There are two possibilities:
   (i) If $|S \cap B_l| \leq b_l$, or if $|S \cap B_l| > b_l$ and $v = v_{l,j}$ for some $j \leq b_l$, then we can set $\xi_v^* = 1$, since doing so does not violate any budget constraint (i.e., $\xi^* \in \Xi$ is satisfied) and does not decrease the value of $\texttt{BPP}(S, \xi^*)$.
   (ii) If $|S \cap B_l| > b_l$ and $v = v_{l,j}$ for some $j > b_l$, then we must have $\xi_v^* = 0$. Indeed, if $\xi_v^* = 1$, and since $\xi^* \in \Xi$, i.e., $\sum_{v' \in B_l} \xi_{v'}^* \leq b_l$, then there exists $k \in \{v_{l,1}, \ldots, v_{l,b_l}\}$ such that $\xi_k^* = 0$. In this case, one can obtain a potentially higher bin packing value associated with the realization obtained by setting $\xi_v^* = 0$ and $\xi_k^* = 1$ (since k has a higher demand with respect to v), contradicting the fact that ξ^* is a maximizer of $\texttt{BPP}(S, \xi)$.

We remark that, with a slight modification, this proof can also be used to show the correctness of the separation procedure for the robust cumulative capacity inequalities outlined in Section 3.4.4.2. □

Proof of Proposition 3.8. The proof is almost identical to that of Proposition 3.4. Nevertheless, we present it for the sake of completeness.

Necessity. Let $S_1(y)$ be any customer order assignment on day 1 such that the existence of a feasible solution $(S_1(y), \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ in $\underline{\mathcal{TSRO}}$ is guaranteed. Also, let $S \subseteq V \setminus \{v \in V : P_v = \{1\}\}$ and $\xi \in \Xi$ be given. Then, we have that:

$$
\begin{aligned}
m(h-1) + \sum_{i \in S} y_{i1} &= m(h-1) + \sum_{i \in S} \mathbb{I}_{[i \in S_1(y)]} \\
&= m(h-1) + |S \cap S_1(y)| \\
&\geq \texttt{BPPDW}(V \setminus S_1(y), \xi) + |S \cap S_1(y)| \\
&\geq \texttt{BPPDW}(S \setminus S_1(y), \xi) + |S \cap S_1(y)| \\
&\geq \texttt{BPPDW}(S, \xi)
\end{aligned}
$$

The first equality follows from (3.17) and the fact that the assignment, $S_1(y)$, is common for realizations ξ and $\hat{\xi}$. The first inequality follows from the following two observations: *(i)* by definition, the customer order assignment $(S_1(y), \tilde{S}_2(\xi), \ldots, \tilde{S}_h(\xi))$ is such that each subset $\tilde{S}_p(\xi)$ for $p > 1$ can be partitioned into m capacity-feasible routes, say $(r_{p,1}, \ldots, r_{p,m})$ and *(ii)* $(S_1(y), \tilde{S}_2(\xi), \ldots, \tilde{S}_h(\xi))$ partitions V. Therefore, the feasible space of the corresponding bin packing problem with day windows defined over the set of items $V \setminus S_1(y)$, under realization ξ, contains the partition $r_{2,1}, \ldots, r_{2,m}, \ldots, r_{h,1}, \ldots, r_{h,m}$. Hence, $m(h-1)$ constitutes an upper bound to its optimal value, resulting in the first inequality. The second inequality follows from the fact that $S \subseteq V$ and that the optimal value of the bin packing problem with day windows over a given set of items is always no smaller than its optimal value over any subset of items. The third inequality follows from the fact that if the set of items is enlarged by an additional item, then the optimal value of the bin packing problem can increase by at most one.

Sufficiency. Assume that y satisfies equations (3.9) and the robust cover inequalities (3.18). We must show that the customer order assignment on day 1, $S_1(y)$, is such that there exists a feasible solution $(S_1(y), \tilde{S}_2(\cdot), \ldots, \tilde{S}_h(\cdot))$ in $\underline{\mathcal{TSRO}}$. Let $\xi \in \Xi$ be any customer order realization. We shall construct the feasible solution as follows. By hypothesis, for $S = V \setminus S_1(y)$ and customer order realization ξ, the robust cover inequality (3.18) is satisfied. The left-hand side of this inequality evaluates to $m(h-1)$, whereas the right-hand side is the optimal value of the bin packing problem with day windows defined over the set of items $V \setminus S_1(y)$ with weights $q_i \xi_i$, day windows $P_i \setminus \{1\}$, and set of days $P \setminus \{1\}$ and bin capacity Q. This implies that there is a capacity-feasible partition of the items, say $(r_{2,1}, \ldots, r_{2,m}, \ldots, r_{h,1}, \ldots, r_{h,m})$. Therefore, for each $p \in P \setminus \{1\}$, we construct $\tilde{S}_p(\xi)$ according to $\tilde{S}_p(\xi) = \bigcup_{k \in K} r_{p,k}$. By construction and from (3.9), $(S_1(y), \tilde{S}_2(\xi), \ldots, \tilde{S}_h(\xi))$ partitions V and is capacity-feasible under realization ξ. □

3.10 APPENDIX: IMPROVED COLUMN-AND-CONSTRAINT GENERATION

In this section, we present algorithmic efficiencies to improve the generalized column-and-constraint generation framework [280] for two-stage robust optimization problems with binary recourse decisions. We first present a brief overview of the original framework and then present our improvements. The bilevel optimization problems $\max\{\texttt{BPP}(S,\xi) : \xi \in \Xi\}$ and $\max\{\texttt{BPPDW}(S,\xi) : \xi \in \Xi\}$ arising in the separation of the robust cover inequalities can be interpreted as special cases of the subproblems that arise in the general framework.

3.10.1 *Algorithmic Improvements*

Consider the following two-stage robust optimization problem with first-stage decisions denoted by x and second-stage decisions denoted by z:

$$\min_{x\in\mathcal{X}} c^\top x + \mathcal{R}(x), \quad \text{where } \mathcal{R}(x) = \max_{\xi\in\Xi} R(x,\xi) \tag{3.21}$$
$$\text{and } R(x,\xi) = \min_{z\in\mathcal{Z}} \left\{ d(\xi)^\top z : W(\xi)z \le h(\xi) - T(\xi)x \right\}.$$

Here, $\mathcal{X} \subseteq \mathbb{R}^{N_1}$, $\mathcal{Z} \subseteq \mathbb{R}^{N_2}$ and $\Xi \subseteq \mathbb{R}^N$ are non-empty and bounded mixed-integer linear representable sets, and $d : \Xi \mapsto \mathbb{R}^{N_2}$, $T : \Xi \mapsto \mathbb{R}^{M\times N_1}$, $W : \Xi \mapsto \mathbb{R}^{M\times N_2}$ and $h : \Xi \mapsto \mathbb{R}^M$ are affine functions. The basic column-and-constraint generation algorithmic framework can be described as follows [280]:

1. Initialize $LB \leftarrow -\infty$, $UB \leftarrow +\infty$ and $L \leftarrow 0$.
2. Let $\left(x^*, \eta^*, \{z^{(l)*}\}_{l=1}^L\right)$ denote the optimal solution of the following problem:

$$\begin{aligned}
&\underset{x,\eta,\{z^{(l)}\}_{l=1}^L}{\text{minimize}} \quad c^\top x + \eta \\
&\text{subject to} \quad x \in \mathcal{X}, \eta \in \mathbb{R} \\
&\qquad \left.\begin{aligned} & z^{(l)} \in \mathcal{Z} \\ & \eta \ge d\left(\xi^{(l)}\right)^\top z^{(l)} \\ & T\left(\xi^{(l)}\right)x + W\left(\xi^{(l)}\right)z^{(l)} \le h\left(\xi^{(l)}\right) \end{aligned}\right\} \forall\, l \in \{1,\dots,L\}
\end{aligned}$$

   Update $LB \leftarrow c^\top x^* + \eta^*$.
   If $UB - LB \le \epsilon$, stop. x^* is the optimal solution of (3.21).
3. Solve $\mathcal{R}(x^*)$ and let ξ^* denote its optimal solution.
   Update $UB \leftarrow \min\{UB, c^\top x^* + \mathcal{R}(x^*)\}$.
   If $UB - LB \le \epsilon$, stop. x^* is the optimal solution of (3.21).
4. Update $L \leftarrow L + 1$. Set $\xi^{(L)} \leftarrow \xi^*$ and go to step 2.

Zhao and Zeng [280] show that the above procedure converges in finite time whenever $R : \mathcal{X} \times \Xi \mapsto \mathbb{R}$ is quasi-convex over Ξ for any $x \in \mathcal{X}$. The most important step in the above procedure is the calculation of $\mathcal{R}(x^*)$ in step 3. Zhao and Zeng [280] propose to do it via another column-and-constraint generation procedure motivated by a reformulation of $\mathcal{R}(x)$ into a trilevel optimization problem. Their procedure to calculate $\mathcal{R}(x)$ makes the following assumptions: (i) *fixed recourse*, i.e., $W(\xi) = W'$ for all $\xi \in \Xi$, and (ii) *complete recourse*, i.e., $R(x, \xi)$ is feasible for any $x \in \mathcal{X}$ and $\xi \in \Xi$. Moreover, it involves a computationally difficult reformulation of $\mathcal{R}(x)$ into a mathematical program with complementarity constraints. In the following, we propose an improved column-and-constraint generation method to calculate $\mathcal{R}(x)$ via a sequence of feasibility problems that alleviates these shortcomings.

Our procedure takes as input a target value d_0 and either returns a scenario $\tilde{\xi} \in \Xi$ for which the recourse cost is higher than d_0; that is, $R(x, \tilde{\xi}) > d_0$, or announces that d_0 is an upper bound to the worst-case recourse cost; that is, $\mathcal{R}(x) \leq d_0$. We can then locate an ϵ-optimal solution of $\mathcal{R}(x)$ by performing a binary search in the space of objective values of $\mathcal{R}(x)$. In doing so, the scenarios identified in previous iterations can be kept in memory in order to speed up convergence. In cases where $\mathcal{R}(x)$ is a feasibility problem (that is, $d \equiv 0$), we do not need to perform a binary search, as our procedure shall either return a scenario $\tilde{\xi} \in \Xi$ for which the recourse problem $R(x, \tilde{\xi})$ is infeasible, or announce that a feasible second-stage decision can be found under any scenario $\xi \in \Xi$.

Our procedure can be described as follows. In the following, $\mathcal{M} \coloneqq \{1, \ldots, M\}$ denotes the index set of second-stage constraints for ease of presentation.

1. Assume $d_0 \in \mathbb{R}$, $\xi^{(0)} \in \Xi$ and $n_{\max} \in \mathbb{N}$ are given.
   Initialize $r \leftarrow 0$, $L \leftarrow 1$ and $\Xi^{(L)} \leftarrow \{\xi^{(r)}\}$.
2. Let $\left(z^{(L)}, s^{(L)}\right)$ denote the optimal solution of the following feasible problem:

$$\begin{aligned} \underset{z,s}{\text{minimize}} \quad & \mathbf{1}^\top s \\ \text{subject to} \quad & z \in \mathcal{Z}, s \in \mathbb{R}^{M+1} \\ & \left.\begin{aligned} d(\xi)^\top z - s_{M+1} &\leq d_0 \\ W(\xi) z - s &\leq -T(\xi)x + h(\xi) \end{aligned}\right\} \forall\, \xi \in \Xi^{(L)} \end{aligned}$$

   If $\mathbf{1}^\top s^{(L)} > 0$, go to step 3. Otherwise, go to step 5.
3. If $\left|\Xi^{(L)}\right| = 1$, stop. $R(x, \xi^{(r)}) > d_0$ is satisfied.
4. Update $\Xi^{(L)} \leftarrow \Xi^{(L)} \setminus \{\xi^{(r)}\}$ and $L \leftarrow L + 1$. Initialize $\Xi^{(L)} \leftarrow \{\xi^{(r)}\}$ and go to step 2.

5. Let $\left(\xi^*, \theta^*, \{\zeta^{(l)*}\}_{l=1}^{L}\right)$ denote the optimal solution of the following problem:

$$\begin{aligned}
&\underset{\xi,\theta,\{\zeta^{(l)}\}_{l=1}^{L}}{\text{maximize}} \quad \theta \\
&\text{subject to} \quad \xi \in \Xi, \theta \in \mathbb{R} \\
&\qquad \left.\begin{aligned}
&\zeta_j^{(l)} \in \{0,1\} \quad \forall\, j \in \mathcal{M} \cup \{0\} \\
&\zeta_j^{(l)} = 1 \Rightarrow \theta \leq \left[T(\xi)\, x + W(\xi)\, y^{(l)}\right]_j - [h(\xi)]_j \;\; \forall\, j \in \mathcal{M} \\
&\zeta_0^{(l)} = 1 \Rightarrow \theta \leq d(\xi)^\top y^{(l)} - d_0 \\
&\zeta_0^{(l)} + \sum_{j=1}^{M} \zeta_j^{(l)} = 1
\end{aligned}\right\} \forall\, l \in \{1, \ldots, L\}
\end{aligned}$$

   If $\theta^* \leq 0$, stop. $\mathcal{R}(x) \leq d_0$ is satisfied.

6. Update $r \leftarrow r+1$ and set $\xi^{(r)} \leftarrow \xi^*$.
   If $\left|\Xi^{(L)}\right| < n_{\max}$, update $\Xi^{(L)} \leftarrow \Xi^{(L)} \cup \{\xi^{(r)}\}$ and go to step 2.
   Otherwise, update $L \leftarrow L+1$, initialize $\Xi^{(L)} \leftarrow \{\xi^{(r)}\}$ and go to step 2.

The key idea behind our above procedure is to enumerate a number of second-stage decisions denoted by $\{z^{(l)}\}_{l=1}^{L}$ that collectively guarantee the feasibility of the second-stage problem $R(x, \xi)$ for any possible scenario $\xi \in \Xi$. The hope is that we would not have to enumerate too many second-stage decisions and that enumerating a few key ones will be sufficient. In the following, we go through the algorithm step by the step.

The procedure is initialized with an initial scenario $\xi^{(0)}$ that is assigned to the scenario subset $\Xi^{(1)}$ (step 1). At iteration L, the subset of scenarios denoted by $\Xi^{(L)}$ is used to generate a second-stage decision $z^{(L)}$, which will ensure that $R(x, \xi) \leq d_0$ is satisfied for all $\xi \in \Xi^{(L)}$ (step 2). Whenever this fails because of the addition of a new scenario (i.e., step 2 yields $\mathbf{1}^\top s^{(L)} > 0$), it implies that either the last added scenario is a certificate of infeasibility (step 3) or we must use the last added scenario to generate an improved second-stage decision (step 4). On the other hand, if we succeed, the optimization problem in step 5 is used to generate a new candidate scenario that makes all currently postulated second-stage decisions $\{z^{(l)}\}_{l=1}^{L}$ infeasible. If no such scenario can be found, we terminate successfully (step 5). Otherwise, we assign this scenario to the subset $\Xi^{(L)}$ that will be used to re-generate an improved second-stage decision $z^{(L)}$ in step 2.

The input parameter $n_{\max}$ controls the growth of the optimization problems in steps 2 and 5 by controlling the maximum cardinality of $\Xi^{(L)}$. The choice $n_{\max} = 1$ would create new variables and constraints in the optimization problem of step 5 every time it is performed, but would restrict the size of the problem in step 2 to involve constraints for just one scenario. On the other hand, $n_{\max} = +\infty$ would create new variables and constraints in the optimization problem of step 5 only when necessary, but would involve constraints for each scenario in $\Xi^{(L)}$. Higher values of $n_{\max}$ are preferable since the

complexity of the optimization problem in step 2 can typically be managed using cutting plane techniques, if necessary.

We remark that the procedure converges in finite time whenever $\mathcal{Z}$ or Ξ are finite sets. We reason as follows. By construction, the separation problem in step 5 can only generate a scenario $\xi^{(r)}$ that makes all currently generated second-stage decisions $\{z^{(l)}\}_{l=1}^{L}$ infeasible. This scenario is then assigned to one of the scenario subsets $\{\Xi^{(l)}\}_{l=1}^{L}$ in a way that ensures it is never identified again in step 5 in subsequent iterations (unless the procedure terminates in step 3). Consequently, no scenario is generated more than once in step 5. Moreover, by the same reasoning, no second-stage decision is generated more than once in step 2 (unless the procedure terminates in step 3).

3.10.2 *Solving* $\max\{\texttt{BPP}(S,\xi) : \xi \in \Xi\}$ *and* $\max\{\texttt{BPPDW}(S,\xi) : \xi \in \Xi\}$

Observe that these problems are special cases of $\mathcal{R}(x)$ in which the recourse problem $R(x,\boldsymbol{\xi})$ is a Bin Packing Problem, and a Bin Packing Problem with Day Windows respectively. Therefore, we can utilize any integer programming formulation of $\texttt{BPP}(S,\xi)$ or $\texttt{BPPDW}(S,\xi)$ to replace $R(x,\xi)$ in the previously described procedure to obtain their maxima. However, we improve numerical tractability in two ways: (i) we utilize an associated feasibility-based integer programming formulation of these problems, and (ii) we do not attempt to obtain the true maxima of these problems.

More specifically, we consider the following integer programing formulation associated with $\texttt{BPP}(S,\xi)$, which either determines a capacity-feasible assignment of items in S to at most m bins, or results in infeasibility implying that the optimal value of the bin packing problem exceeds m. Here, binary variables z_{ik} indicate if item $i \in S$ is assigned to bin $k \in K := \{1,\dots,m\}$.

$$
\begin{aligned}
\underset{z}{\text{minimize}}\ & 0 \\
\text{subject to}\ & z_{ik} \in \{0,1\} && \forall\, i \in S,\ \forall\, k \in K \\
& \sum_{k\in K} z_{ik} = 1 && \forall\, i \in S \\
& \sum_{i\in S} (q_i \xi_i)\, z_{ik} \le Q && \forall\, k \in K \\
& \sum_{j\in S: j<i} z_{j,k-1} \ge z_{ik} && \forall\, i \in S,\ \forall\, k \in K \setminus \{1\}
\end{aligned}
\tag{3.22}
$$

Similarly, we consider the following integer programming formulation associated with $\texttt{BPPDW}(S,\xi)$, which either determines a capacity-feasible and day window-feasible assignment of items in S to at most m bins available on each day $p \in P \setminus \{1\}$, or results in

infeasibility implying that the optimal value exceeds $m(h-1)$. Here, binary variables z_{ikp} indicate if item i is assigned to bin k on day p.

$$
\begin{aligned}
\underset{z}{\text{minimize}}\ & 0 \\
\text{subject to}\ & z_{ikp} \in \{0,1\} && \forall\, i \in S,\ \forall\, k \in K,\ \forall\, p \in P \setminus \{1\} \\
& \sum_{p \in P_i \setminus \{1\}} \sum_{k \in K} z_{ikp} = 1 && \forall\, i \in S \\
& \sum_{i \in S} (q_i \xi_i)\, z_{ik} \leq Q && \forall\, k \in K,\ \forall\, p \in P \setminus \{1\} \\
& \sum_{j \in S: j < i} z_{j,k-1,p} \geq z_{ikp} && \forall\, i \in S,\ \forall\, k \in K \setminus \{1\},\ \forall\, p \in P \setminus \{1\}
\end{aligned}
\tag{3.23}
$$

Note that, in both formulations (3.22) and (3.23), the last constraint is a symmetry-breaking constraint to enforce that, if item i is assigned to bin $k > 1$, then at least one item $j < i$ must be assigned to bin $k-1$.

We now outline the procedure to solve the bilevel problem $\max\{\texttt{BPP}(S,\xi) : \xi \in \Xi\}$ arising in the separation of the robust cover inequalities (3.11). We first replace the second-stage problem $R(x,\xi)$ with formulation (3.22) and use the procedure described in the previous section with input parameters $d_0 = 0$, $\xi^{(0)} \in \arg\max\{\sum_{i \in S} q_i \xi_i : \xi \in \Xi\}$ and $n_{\max} = 50$. This procedure either returns a scenario $\tilde{\xi} \in \Xi$ for which formulation (3.22) is infeasible or certifies that formulation (3.22) is feasible for all $\xi \in \Xi$. The former implies that $\texttt{BPP}(S,\tilde{\xi}) > m$, while the latter implies that $\max\{\texttt{BPP}(S,\xi) : \xi \in \Xi\} \leq m$. In the former case, we obtain the exact value of $\texttt{BPP}(S,\tilde{\xi})$ by solving a deterministic bin packing problem using the exact algorithm MTP described in [196] and use the resulting value when adding the corresponding robust cover inequality (3.11). We remark that, in both outcomes, the exactness of the separation algorithm for the robust cover inequalities (3.11) is guaranteed.

The procedure to solve $\max\{\texttt{BPPDW}(S,\xi) : \xi \in \Xi\}$ arising in the separation of inequalities (3.18) is exactly the same, except that we replace the second-stage problem $R(x,\xi)$ with formulation (3.23). Moreover, in the case when we have identified a scenario $\tilde{\xi} \in \Xi$ for which $\texttt{BPPDW}(S,\tilde{\xi}) > m(h-1)$, we do not compute the exact value of $\texttt{BPPDW}(S,\tilde{\xi})$ but rather use the lower bound of $m(h-1)+1$ when adding the corresponding inequality (3.18), since our computational experience suggested that the optimal value of $\texttt{BPPDW}(S,\tilde{\xi})$ almost never exceeds this lower bound. We remark that this does not invalidate the correctness of the lower bounds provided by formulation $\underline{\mathcal{TSRO}}_{IP}$.

4

TACTICAL ENFORCEMENT OF SERVICE CONSISTENCY

Chapter 2 considered the problem of designing *static* routes to serve customers with uncertain demands over an operational (*e.g.*, one day) horizon. One benefit of static routes are that they are *consistent*: the same driver serves the same set of customers at roughly the same time. Such consistent routes are easy to adapt to the realization of daily uncertainties. Moreover, consistency helps companies realize the important goal of personalization of services while also improving driver productivity and familiarity with their daily routes. In this chapter, we aim to explicitly incorporate service consistency as a means to hedge against demands (as well as service times) that vary over a tactical (*e.g.*, one week) horizon.

To that end, we study the Consistent Traveling Salesman Problem, where the goal is to identify the minimum-cost set of routes that a single vehicle should follow during the multiple time periods of a planning horizon. The requirement for consistent service is defined to be equivalent to restricting the difference between the earliest and latest vehicle arrival-times, across the multiple periods, to not exceed some given allowable limit. We present two exact algorithms for this problem. The first is a branch-and-cut algorithm based on three novel mixed-integer linear programming formulations, while the second is a decomposition algorithm based on decomposing the multi-period problem into a sequence of single-period "classical" traveling salesman problems with time windows. These constitute the first exact algorithms in the open literature that consider consistency constraints.

We show that our algorithms are capable of solving instances whose sizes are representative–or even exceed–expected sizes of real-world distribution settings involving a single vehicle. Moreover, we also empirically show that *(i)* consistency can be achieved with merely a small increase in total routing costs (as compared to the case where consistency considerations are not taken into account), and *(ii)* the cost of implementing consistent routes can be reduced significantly if vehicles are allowed to idle en route.

This chapter is structured as follows. Section 4.1 motivates the problem and provides necessary background information. Section 4.2 formally defines the problem and introduces the necessary notation. Section 4.3 presents the branch-and-cut algorithm, while

Section 4.4 presents the decomposition algorithm. Finally, Section 4.5 presents extensive computational results, comparing the performance of the two algorithms, elucidating the cost of providing consistent service as well as the cost savings that are possible when vehicle idling is allowed to occur.

4.1 BACKGROUND AND MOTIVATION

Supply chains have gradually shifted attention from company-focused to customer-focused operating strategies over the last few years. Under this paradigm, the focus is to improve the quality of service and to stimulate one's profit by satisfying customer needs better than one's competitors. In the context of distribution operations, an important lever to gain this competitive advantage is to provide customer service that is consistent over time. Consistency in service is particularly desirable in the design and implementation of periodic distribution systems, wherein customers require frequent service across multiple time periods. In addition to providing competitive advantages through customer satisfaction, adopting consistent service policies exposes efficiencies that reduce costs for both the distributor and the customer.

According to Kovacs et al. [172], who surveyed periodic vehicle routing problems in which service consistency considerations have been explicitly addressed, consistency in this context translates to satisfying any of the following requirements each time service is provided to a customer: (i) arrival-time consistency, wherein the customer should be visited at roughly the same time, (ii) person-oriented consistency, in which the customer should be visited by the same driver, and (iii) delivery consistency, for which a customer should receive roughly the same quantity of goods. Examples of real life applications where such consistent policies have been applied include–among others–courier services [140, 225, 245, 270], vendor-manged inventory distribution [84, 99], home care and nursing services for the elderly [110, 271] and aircraft fleet scheduling [155].

This work focuses on the aspect of arrival-time consistency. In this context, the supplier aims to reduce the variability in the actual times during the routing horizon at which a customer is served, since doing so generally increases the value of service for the customer. For example, in the context of VMI distribution, arrival-time consistency reduces the need for the customer to commit loading-dock resources throughout the day. In the home-care industry, the elderly and disabled are sensitive to changes in their daily routines and developing consistent schedules is of particular importance. Moreover, from the service provider's point of view, reducing the variability across repetitive deliveries over multiple time periods can expose efficiencies that add up to significant cost savings.

The Consistent Traveling Salesman Problem (ConTSP) is a variant of the well-known Traveling Salesman Problem (TSP) that attempts to address the issue of arrival-time consistency in multi-period routing applications. In the ConTSP, we aim to design minimum-cost (or, minimum-makespan) routes over a finite, multi-period horizon so as to serve a set of customers with known demands and service durations using a single

(uncapacitated) vehicle. In general, a customer may or may not require service in a given time period and, thus, only a subset of customers need to be visited in each period. The consistency requirement applies to every customer who requires service in more than one time period, meaning that every such customer must be visited at roughly the same time in each period for which service is required. The exact time of service remains a decision variable, but the arrival times across the multiple periods at each customer site must not differ by more than some prespecified, constant bound, which we call the maximum allowable arrival-time differential. In practice, this bound may be set by either the customer or by the service provider. Note that, if the maximum allowable arrival-time differential is chosen to be equal to infinity, then the ConTSP reduces to a set of separable TSPs (one for each time period); hence, the ConTSP is $\mathcal{NP}$–hard, just like the TSP [166].

We mention here papers in the existing literature that study problems closely related to ours. The Periodic Vehicle Routing Problem [74, 233] also addresses periodicity in routing applications. In this problem, each customer requires one or more visits over a planning horizon and the distributor must select in which time periods to provide these visits, while minimizing the total cost across the planning horizon. Spliet and Gabor [242], Spliet and Desaulniers [241], and Jabali et al. [157] investigate the problem of assigning a single time window to each customer before the start of the planning horizon such that the service provider attempts to meet these time windows on a daily basis and minimizes the expected cost under operational uncertainty. These endogenous time windows are motivated by requirements of arrival-time consistency and they serve the same purpose as the above-mentioned maximum allowable arrival-time differential in the ConTSP. Finally, Kovacs et al. [172] remark that the problem of visiting customers at consistent times over the different periods in a planning horizon is similar to single-period, multi-vehicle routing problems with temporal synchronization of vehicles [106]. In particular, consistent routes in a multi-period setting are equivalent to several synchronized vehicles that must arrive to customer sites almost simultaneously during a single-period; the solution of the latter problem can be recovered as the union of all single-vehicle routes in the former problem.

We highlight that the ConTSP constitutes a special, single-vehicle case of the Consistent Vehicle Routing Problem (ConVRP), originally introduced in [140]. The ConVRP utilizes multiple capacitated vehicles in order to provide consistent service to a set of customers over multiple periods while minimizing total transportation cost. In addition to considering arrival-time consistency, the ConVRP also requires driver consistency, for which each customer must be visited by the same driver in all periods of the planning horizon. A number of metaheuristic approaches to solve the ConVRP have been proposed in the literature. Groër, Golden, and Wasil [140] present an algorithm that is based on generating a "template" route by following a precedence principle: if customer a is visited before customer b in a time period, then a should be visited before b in every time period. The template route is built by considering only customers that require service in more than one time period. The individual routes in each time period are then generated

by deleting those customers in the template who do not require service in that period, while those customers that require service in this period but are not part of the template route are inserted into the solution at the best position. Kovacs, Parragh, and Hartl [171] build upon this algorithm by allowing deviations from the precedence template using an adaptive large neighborhood search procedure; Tarantilis, Stavropoulou, and Repoussis [249] modify both the template route and the actual single period routes using tabu search. Note that, in all of the above approaches, waiting is not allowed; that is, the vehicle is not allowed to wait at a customer location before providing service. Moreover, in [140, 249], the departure times of the vehicles from the depot are fixed: vehicles must depart from the depot at time zero in each period. Kovacs, Parragh, and Hartl [171] relax the latter requirement by allowing the vehicle to delay its departure from the depot and demonstrate that service quality can be improved without increasing driver working time. In a more recent paper, Kovacs et al. [173] consider a generalization of the ConVRP in which a limited number of different drivers may visit a customer, the maximum difference in the earliest and latest arrival times is penalized in the objective (i.e., arrival-time consistency is treated as a soft requirement), and each customer is associated with either of two available time windows (e.g., AM or PM). Finally, Luo et al. [192] study a multi-period variant of the Vehicle Routing Problem with Time Windows, wherein each customer may be visited by only a limited number of drivers over the routing horizon, in addition to being serviced within a given time window.

It should be highlighted that, in all of the above approaches, vehicle idling (a.k.a., "waiting") is not allowed; that is, the vehicle is not allowed to wait at a customer location before providing service.[1] In [171, 173], the former requirement is slightly relaxed by allowing the vehicle to delay its departure from the depot[2] and show that service quality can be improved without increasing driver working time. Moreover, with some of the existing approaches, it is not always straightforward to incorporate maximum limits on total travel time that are motivated by hours-of-work regulations in many contexts.

In this chapter, we present two new algorithms to address the ConTSP. To the best of our knowledge, this is the first attempt in the open literature to address a routing problem with consistency requirements in an exact framework. Our motivation to address the special, single-vehicle case of the ConVRP is three-fold:

(i) an exact approach, even if only applicable to single-vehicle instances, has the potential to provide higher quality solutions with provable guarantee of optimality for those instances,

(ii) if the assignment of customers to drivers is fixed, as is commonly done via districting or sectoring in territory-based planning [172, 225, 233], then the ConVRP decomposes into several ConTSPs, one for each driver; consequently, an efficient so-

1 Vehicle idling does not necessarily entail actual waiting at the customer premises. Idling may be alternatively implemented by slowing the vehicle down to an appropriate speed as it approaches the customer site.

2 Delaying departure from the depot is equivalent to allowing waiting only at the location of whichever (solution-dependent) customer is visited first in the route.

lution scheme for the ConTSP could serve as a component of a hybrid metaheuristic or a decomposition-based exact approach for the ConVRP.

(iii) our contribution can influence the development of new exact algorithms to address more complex problem settings for consistent routing.

We note that the size of instances (100 customers over a 5-period horizon) we are able to address are representative–or even exceed–expected sizes of real world distribution settings involving a single vehicle. We also relax the restrictive assumption that vehicle waiting is not allowed, allowing the vehicle to wait at any customer location before starting service, in addition to enabling the consideration of maximum route duration limits, as part of our new solution approach. These generalizations are representative of real world routing operations and alleviate existing algorithmic assumptions that may be too restrictive in practice. We remark that our methods can readily incorporate the AM/PM time window requirements proposed in [173] or the delayed vehicle departure times proposed in [171, 173].

4.2 PROBLEM DEFINITION

Let $G = (V, A)$ be the complete directed graph on $n+1$ nodes, where $V := \{0, 1, \ldots, n\}$ is the node set and $A := \{(i, j) \in V \times V : i \neq j\}$ is the arc set. Node 0 represents the depot, while the node subset $V_c = V \setminus \{0\}$ represents the set of customers. Associated with every arc $(i, j) \in A$ is a travel cost $c_{ij} \geq 0$ and a travel duration $t_{ij} \geq 0$. Service (a.k.a., processing) times can be easily incorporated into the travel durations via the operation $t_{ij} \leftarrow t_{ij} + s_i$, where $s_i \geq 0$ is the service time of each customer $i \in V_c$ and $s_0 = 0$. We remark that we do not require symmetric costs or travel times; that is, we allow $c_{ij} \neq c_{ji}$ or $t_{ij} \neq t_{ji}$ for any $(i, j) \in A$. In either case, since in general service times differ among nodes, the ConTSP constitutes an asymmetric problem even if the costs and travel times are themselves symmetric.

Let also $\mathcal{P} = \{1, \ldots, h\}$ denote the set of time periods such that $h \geq 2$. For each period $p \in \mathcal{P}$, we define the node subset $V_p \subseteq V_c$ to be the set of customers requiring service in this period, and we define the associated subset of arcs to be $A_p := \big\{(i, j) \in \big(V_p \cup \{0\}\big) \times \big(V_p \cup \{0\}\big) : i \neq j\big\}$. Without loss of generality, we assume for each period $p \in \mathcal{P}$, that $V_p \cap \big[\cup_{q \in \mathcal{P}: q \neq p} V_q\big] \neq \emptyset$; that is, each set V_p is non-empty[3] and it in fact includes at least one customer requiring service in some additional period.[4] For each customer $i \in V_c$, we let $\mathcal{P}_i := \{p \in \mathcal{P} : i \in V_p\}$ denote the set of time periods in which customer i requires service.

3 If the input data provides for a period in which no customers require service, then that period can be removed from consideration.

4 If the input data provides for a time period p in which every customer that requires service in that period does not require service in any additional period $q \neq p$, then we can independently address period p by solving its corresponding TSP instance and appending its solution to the solution of the ConTSP instance induced by the remaining periods.

A traveling salesman tour (TSP tour) in period p is a Hamiltonian cycle in $G_p := (V_p \cup \{0\}, A_p)$. We refer to such a tour via notation $T_p = \langle v_{p,0} = 0, v_{p,1}, v_{p,2}, \ldots, v_{p,|V_p|}, 0 \rangle$, where it is implied that each customer $v_{p,k}$ is unique. Given a tour T_p, we define its cost as $c(T_p) := \sum_{k=1}^{|V_p|} c_{v_{p,k-1}v_{p,k}} + c_{v_{p,|V_p|}0}$. Furthermore, we can define the arrival time, $a^p_{v_{p,k}}$, at the k^{th} node, $v_{p,k} \in V_p$, as any feasible solution to the linear system (4.1). A TSP tour is said to be infeasible if there is no feasible solution to this linear system.

$$a^p_{v_{p,0}} = 0 \tag{4.1a}$$

$$a^p_{v_{p,k}} \geq a^p_{v_{p,k-1}} + t_{v_{p,k-1}v_{p,k}} \qquad \forall\, k \in \{1, \ldots, |V_p|\} \tag{4.1b}$$

$$a_{v_{p,|V_p|}} + t_{v_{p,|V_p|}0} \leq D \tag{4.1c}$$

Here, D is the allowable route duration limit, i.e., an upper bound on the total time elapsed until the vehicle returns to the depot (if no such restriction applies, a sufficiently large value may be used instead). Note that the above definition allows the vehicle to wait at customer locations before starting service. If the vehicle is not allowed to wait, then the inequality in (4.1b) must be replaced with an equality.

Furthermore, if time windows also apply, then the resulting tour is called a traveling salesman tour with time windows (TSPTW tour). More specifically, if $\mathcal{TW}_i = [e_i, \ell_i]$ is the time window for customer $i \in V_p$, where $e_i \geq 0$ is the earliest arrival time and $\ell_i \geq e_i$ is the latest arrival time for customer i, then the arrival times can be redefined as any feasible solution to the above linear system with the added restriction $e_i \leq a^p_i \leq \ell_i$ (also denoted as $a^p_i \in \mathcal{TW}_i$) and the corresponding tour T_p is said to be feasible only if there exists a feasible solution to the augmented linear system. Note that a feasible TSP tour is also a feasible TSPTW tour whenever the time windows have been relaxed for each customer, i.e., whenever $\mathcal{TW}_i = [0, D]$ for all $i \in V_c$.

A ConTSP solution is a collection of TSP tours $\{T_1, T_2, \ldots, T_{|\mathcal{P}|}\}$, i.e., one tour for each period $p \in \mathcal{P}$. Given such a collection and a corresponding feasible solution to the linear system (4.1) for each period $p \in \mathcal{P}$, we define $\Delta a^{\max}_i := \max_{p \in \mathcal{P}_i} a^p_i - \min_{p \in \mathcal{P}_i} a^p_i$ to be the arrival-time differential for a customer i. In the ConTSP, we want to enforce that this arrival-time differential does not exceed the given maximum allowable value, $L > 0$, for all customers $i \in V_c$; that is, a ConTSP solution is feasible if and only if there exists a feasible solution to the linear system (4.1) that satisfies $\max_{i \in V_c} \Delta a^{\max}_i \leq L$. A collection of tours is said to be consistent if they induce a feasible ConTSP solution. The objective of the ConTSP is then to determine the collection of consistent tours with the minimum sum of costs, $\sum_{p \in \mathcal{P}} c(T_p)$.

4.3 BRANCH-AND-CUT ALGORITHM

In this section, we present a branch-and-cut algorithm for the ConTSP. We note that this algorithm is applicable only when vehicle idling is not permitted, *i.e.*, if we impose $a_i = 0$ for all $i \in V_p$ in (4.1).

In Section 4.3.1, we present three alternative mixed-integer linear programming formulations and compare their performance when serving as the basis of our branch-and-cut scheme. In Section 4.3.2, we present a number of valid inequalities for the above formulations. More specifically, in our implementation we use Subtour Elimination Constraints and 2-matching Constraints. We further derive a new class of valid inequalities for the ConTSP, which we refer to as Inconsistent Path Elimination Constraints. In Section 4.3.3, we discuss the implementation of our branch-and-cut algorithm, including separation routines for each of the valid inequalities.

Throughout this section, we use the following notation. For each customer $i \in V_p \cup \{0\}$, let $N_p^+(i)$ denote the set of nodes j for which there is an arc from i to j in the graph G_p, i.e., $N_p^+(i) := \{j \in V : (i,j) \in A_p\}$. Similarly, let $N_p^-(i) := \{j \in V : (j,i) \in A_p\}$. Finally, given a subset of nodes $S \subseteq V_p \cup \{0\}$, let $A(S)$ be the set of arcs with both end points in S, i.e., $A(S) := \{(i,j) \in S \times S : i \neq j\}$, and let $\delta(S)$ be the set of arcs with exactly one end point in S, i.e., $\delta(S) := \{(i,j) \in S \times S^c\} \cup \{(i,j) \in S^c \times S\}$, where $S^c = \big(V_p \cup \{0\}\big) \setminus S$.

4.3.1 *Formulations*

We now present three alternative mixed-integer linear programming formulations that can serve as the basis for a branch-and-cut approach. These formulations differ in how they encode the various useful quantities (e.g., the arrival times at customer locations) and how they enforce the ConTSP's requirements, namely the requirement that each period's tour corresponds to a Hamiltonian tour (i.e., each node is incident to one outgoing and one incoming arc and there are no subtours) as well as that the tours across all periods are consistent (in the sense described in Section 4.2). Sometimes, a requirement is described via a superpolynomial set of constraints, which will have to be added dynamically in the context of a branch-and-cut framework. In the following, we will discuss the various strategies that can be followed in each case with regards to which constraints are to be utilized as cutting planes.

Let binary variables x_{ijp} be defined as follows:

$$x_{ijp} = \begin{cases} 1, & \text{if arc } (i,j) \in A_p \text{ is used in the tour of period } p \\ 0, & \text{otherwise} \end{cases}$$

Using these variables, the ConTSP can be cast with the following conceptual formulation:

$$\min \sum_{p \in \mathcal{P}} \sum_{(i,j) \in A_p} c_{ij} x_{ijp} \tag{4.2}$$

$$\text{s.t.} \quad x_{ijp} \in \{0,1\} \qquad \forall\ (i,j) \in A_p,\ \forall\ p \in \mathcal{P} \tag{4.3}$$

$$\sum_{j \in N_p^+(i)} x_{ijp} = 1 \qquad \forall\, i \in V_p \cup \{0\}\,,\ \forall\ p \in \mathcal{P} \tag{4.4}$$

$$\sum_{j \in N_p^-(i)} x_{jip} = 1 \qquad \forall\, i \in V_p \cup \{0\}\,,\ \forall\ p \in \mathcal{P} \tag{4.5}$$

$$\left\{(i,j) \in A_p : x_{ijp} = 1\right\} = T_p \qquad \forall\ p \in \mathcal{P} \tag{4.6}$$

$$\left\{T_p, T_q\right\} \text{ is consistent} \qquad \forall\ (p,q) \in \mathcal{P} \times \mathcal{P} : p < q \tag{4.7}$$

In this formulation, the assignment (degree) constraints (4.4) and (4.5) ensure that each node is incident to one outgoing and one incoming arc in each period where it appears. Constraints (4.6) eliminate subtours by imposing that only Hamiltonian circuits be considered in each time period. Finally, constraints (4.7) ensure that the tours across all periods are consistent (in the sense described in Section 4.2).

The formulations we are about to present below differ in the way they achieve the requirements of constraints (4.6) and (4.7). The first formulation involves only the binary arc variables x_{ijp} defined above and utilizes a superpolynomial number of constraints to eliminate subtours and enforce consistency. In addition to binary arc variables x_{ijp}, the second formulation utilizes continuous node variables α_{ip} to represent the arrival time at customer i in period p, and employs big-M constraints to appropriately encode them. The use of big-M constraints is a common way to encode arrival times in formulations for the Vehicle Routing Problem with Time Windows (see [103]). The third formulation uses continuous arc variables y_{ijp} in addition to binary variables x_{ijp} and is based on a single-commodity flow representation of travel time.

4.3.1.1 *Formulation 1*

This model does not utilize any additional variables beyond the binary variables x_{ijp} defined above. Constraints (4.6) are modeled via Eqs. (4.8), which constitute the standard Subtour Elimination Constraints (SEC) originally proposed by Dantzig, Fulkerson, and Johnson [96].

$$\sum_{(i,j) \in A(S)} x_{ijp} \leq |S| - 1 \quad \forall\ S \subseteq V_p \cup \{0\} : 2 \leq |S| \leq \left|V_p\right| - 1,\ \forall\ p \in \mathcal{P} \tag{4.8}$$

The consistency constraints (4.7) are modeled using a new class of infeasible path elimination constraints [16], which we introduce in this work and refer to as Inconsistent Path Elimination Constraints (IPEC). Note that two paths in two different periods are said to be "inconsistent" if their simultaneous occurrence as sub-paths in the tours of their respective time periods renders the ConTSP solution infeasible, and each IPEC will forbid such an occurrence for a pair of inconsistent paths.

Let $P = (v_1, \ldots, v_k)^p$ denote a non-empty path in period p that is formed by the arcs in the set $\{(v_i, v_{i+1}) : i = 1, \ldots, k-1\}$, where $(v_i, v_{i+1}) \subseteq A_p$ for all $i = 1, \ldots, k-1$.

We assume this path to be open and simple, i.e., $k > 1$ and $v_i \neq v_j$ for $i \neq j$, and we use $|P| = k - 1$ to denote the path length, which is equal to the cardinality of the above arc set. Let also $\tau(P) = \sum_{i=1}^{k-1} t_{v_i v_{i+1}}$ denote the time spent traveling on this path and processing all its nodes except the last one, v_k. Given these definition, sufficient conditions for inconsistency of a pair of paths are given in Lemma 4.1.

Lemma 4.1. *A pair of paths $\{P, Q\}$, where $P = (v_1, \ldots, v_s)^p$, $Q = (w_1, \ldots, w_t)^q$ and $p \neq q$, is inconsistent, if any of the following conditions holds:*

(i) $w_1 = v_1 = 0$, $w_t = v_s$ *and* $|\tau(P) - \tau(Q)| > L$

(ii) $w_1 = v_1 \neq 0$, $w_t = v_s \neq 0$ *and* $|\tau(P) - \tau(Q)| > 2L$

(iii) $w_1 = v_s \neq 0$, $w_t = v_1 \neq 0$ *and* $\tau(P) + \tau(Q) > 2L$

Proof. Consider a ConTSP solution in which P and Q appear as sub-paths in time periods p and q respectively. Denote by a_c^p the arrival time at customer c in period p. It holds from their definitions that $\tau(P) = a_{v_s}^p - a_{v_1}^p$ and $\tau(Q) = a_{w_t}^q - a_{w_1}^q$. Let us take into account these relationships in the context described by each of the three conditions:

(i) Since $w_t = v_s$, we have $a_{w_t}^q = a_{v_s}^q$. Furthermore, since the vehicle always departs from the depot at time o, we have $a_{v_1}^p = a_{w_1}^q = 0$. Therefore, the condition $|\tau(P) - \tau(Q)| > L$ implies that $\left|a_{v_s}^p - a_{v_s}^q\right| > L$; that is, customer v_s violates the maximum allowable arrival-time differential, rendering the pair of paths $\{P, Q\}$ to be inconsistent.

(ii) Since $w_1 = v_1$ and $w_t = v_s$, we have $a_{w_1}^q = a_{v_1}^q$ and $a_{w_t}^q = a_{v_s}^q$. Therefore, the condition $|\tau(P) - \tau(Q)| > 2L$ implies that $\left|a_{v_s}^p - a_{v_1}^p - a_{v_s}^q + a_{v_1}^q\right| > 2L$, which in turn implies that either $\left|a_{v_1}^q - a_{v_1}^p\right| > L$ or $\left|a_{v_s}^p - a_{v_s}^q\right| > L$ (or both); that is, at least one of the two customers v_1 and v_s violates the maximum allowable arrival-time differential, rendering the pair of paths $\{P, Q\}$ to be inconsistent.

(iii) Since $w_1 = v_s$ and $w_t = v_1$, we have $a_{w_1}^q = a_{v_s}^q$ and $a_{w_t}^q = a_{v_1}^q$. Therefore, the condition $\tau(P) + \tau(Q) > 2L$ implies that $\left(a_{v_s}^p - a_{v_1}^p + a_{v_1}^q - a_{v_s}^q\right) > 2L$, which in turn implies that $\left|a_{v_s}^p - a_{v_1}^p + a_{v_1}^q - a_{v_s}^q\right| > 2L$; that is, the same conclusion as in case (ii) above can be reached.

□

Lemma 4.1 states that if P and Q have common end nodes and if one of them, say v_1, is to be visited at consistent times in periods p and q, then the other common end node, v_s, cannot be visited at consistent times in those time periods, if the travel times to v_s along paths P and Q are sufficiently different.

The basic form of the inequality that forbids the simultaneous occurrence of P and Q in a solution is presented in Eq. (4.9).

$$\sum_{(i,j)\in P} x_{ijp} + \sum_{(i,j)\in Q} x_{ijq} \leq |P| + |Q| - 1 \tag{4.9}$$

Every feasible ConTSP solution must satisfy constraints (4.9) for every inconsistent pair of paths $\{P, Q\}$. Moreover, every solution to the set of constraints (4.3)–(4.5) and (4.8) (i.e., a set of tours) that violates the arrival-time consistency requirement for at least one customer must satisfy condition (i) of Lemma 4.1 for at least one pair of paths. Therefore, every solution that satisfies the set of constraints (4.3)–(4.5), (4.8) and (4.9), where the latter is imposed for all possible pairs of paths that meet the conditions of Lemma 4.1, constitutes a feasible ConTSP solution. As a result, the model consisting of Eqs. (4.2)–(4.5), (4.8) and (4.9) is a complete and valid ConTSP formulation. Note that this formulation is similar to the TSP with Time Windows (TSPTW) formulation described in [17], which also consisted of only binary arc variables.

We remark that the number of distinct SEC is $\mathcal{O}\left(2^n h\right)$. Similarly, the number of distinct IPEC in the worst case, when every possible pair of paths is inconsistent, is $\mathcal{O}\left(n!^2 h^2\right)$. Therefore, since the number of these constraints grows very fast (exponentially and factorially, respectively) with the size of the instance, we treat these inequalities as cutting planes and add them dynamically in a branch-and-cut solution framework. In practice, the number of such inequalities added is relatively small. Our separation procedures and associated separation protocols are discussed in detail in Section 4.3.3. Note that, although it may be more challenging in the case of a fractional solution, one can immediately and exactly (i.e., with guarantees to identify a violation, if one exists) separate either of these inequalities from an integral solution (see Section 4.3.3 for details). To that end, the branch-and-cut framework is guaranteed to locate the optimal solution, as long as it is afforded enough computational resources. We discuss our computational experience in Section 4.5.

4.3.1.2 *Formulation 2*

This formulation explicitly encodes the arrival times at customers. To that purpose, we introduce continuous variables $\alpha_{ip} \geq 0$ to capture the arrival time at each customer $i \in V_p$ in each period $p \in \mathcal{P}$. These variables attain appropriate values via their participation in Miller-Tucker-Zemlin (MTZ) expressions [198], which are cast here in terms of travel time.[5] For each $p \in \mathcal{P}$, let us define parameters $f_{ip} := \min_{k \in N_p^+(i)} t_{ik}$, $r_{ip} := \min_{k \in N_p^-(i)} t_{ki}$, for all $i \in V_p$ and $f_{0p} := 0$, $r_{0p} := 0$; also define $\xi_{ip} := \max_{j \in N_p^+(i)} \left\{t_{ij} + f_{jp}\right\}$, for all $i \in V_p$. Constraints (4.10) apply.

$$\alpha_{ip} - \alpha_{jp} + t_{ij} x_{ijp} - t_{ji} x_{jip} \leq \left(M_p - f_{ip} - r_{jp}\right)\left(1 - x_{ijp} - x_{jip}\right) \quad \forall\, (i,j) \in A\left(V_p\right),\ \forall p \in \mathcal{P} \tag{4.10a}$$

$$\alpha_{ip} \geq \sum_{j \in N_p^-(i)} \left(r_{jp} + t_{ji}\right) x_{jip} \quad \forall i \in V_p,\ \forall p \in \mathcal{P} \tag{4.10b}$$

5 In the typical setting, MTZ expressions are cast in terms of order of customer visits or cumulative demand served [103].

$$\alpha_{ip} + \sum_{j \in N_p^+(i)} \left(t_{ij} + f_{jp}\right) x_{ijp} \leq M_p \left(1 - x_{0ip}\right) + \left(t_{0i} + \xi_{ip}\right) x_{0ip} \quad \forall i \in V_p,\ \forall p \in \mathcal{P} \tag{4.10c}$$

$$\alpha_{ip} \leq \left(M_p - f_{ip}\right) \left(1 - x_{0ip}\right) + t_{0i} x_{0ip} \quad \forall i \in V_p,\ \forall p \in \mathcal{P} \tag{4.10d}$$

The parameters M_p are big-M coefficients that must be large enough so as not to exclude the optimal solution. If z_{UB} is the objective value of a known feasible solution, obtained via a heuristic or otherwise, and if ζ_p^* is the optimal objective of the TSP in period p, then the value $M_p = z_{\text{UB}} - \sum_{q \in \mathcal{P}: q \neq p} \zeta_q^*$ suffices.[6] This also implies that the value of the parameter M_p can be dynamically tightened during the search as new incumbent solutions are identified.

We remark that the MTZ expressions as presented in Eqs. (4.10) incorporate several applicable liftings, which we have proposed here for the first time. We further remark that, if the triangle inequality on the travel time vector t is satisfied, namely if

$$t_{ij} + t_{jk} \geq t_{ik} \quad \forall\ i, j, k \in V \tag{4.11}$$

then one may further lift the formulation by updating the definitions of its parameters as follows: $f_{ip} := t_{i0}$, $r_{ip} := t_{0i}$, and $\xi_{ip} := \max_{j \in N_p^+(i)} \left\{t_{ij} + t_{j0}\right\}$.

Constraints (4.10) suffice to exclude subtours.[7] Furthermore, the introduction of variables α_{ip} enables the explicit enforcement of arrival-time consistency via constraints (4.12), achieving the arrival-time consistency requirement (4.7).

$$\alpha_{ip} - \alpha_{iq} \leq L \quad \forall\, i \in V_p \cap V_q,\ \forall\, (p, q) \in \mathcal{P} \times \mathcal{P} : p \neq q \tag{4.12}$$

Therefore, the model consisting of Eqs. (4.2)–(4.5), (4.10) and (4.12) is a complete and valid ConTSP formulation.

4.3.1.3 *Formulation 3*

This is a single-commodity flow formulation, where the cumulative travel time is represented as a commodity, originating at the depot and flowing through the arcs. The single-commodity flow formulation for the TSP was originally proposed in [124], where one unit of commodity was picked up along each traveled arc. A similar model was proposed in [194] for the sequential ordering problem with time windows; the version presented here can be obtained from the latter by ignoring the precedence constraints.

6 If no feasible solution to the problem is known, one may replace z_{UB} with the sum of the optimal values of the Maximum TSPs in each period.

7 On a rather technical remark, note that MTZ constraints suffice to exclude a potential subtour only when the total travel time along this subtour is strictly positive (which is the typical case). For data sets that involve arcs of zero travel time, which may give rise to subtours of zero total travel time, one should pay attention to consider the relevant SEC explicitly (e.g., by adding them as a lazy cut).

After introducing a set of continuous variables $y_{ijp} \geq 0$ to capture the commodity flow on each arc $(i,j) \in A_p$ in each period $p \in \mathcal{P}$, the following constraints apply.

$$\sum_{k \in N_p^+(j)} y_{jkp} = \sum_{i \in N_p^-(j)} \left(y_{ijp} + t_{ij} x_{ijp} \right) \quad \forall \ j \in V_p, \ \forall \ p \in \mathcal{P} \tag{4.13a}$$

$$\sum_{j \in N_p^+(0)} y_{0jp} = 0 \quad \forall \ p \in \mathcal{P} \tag{4.13b}$$

$$0 \leq y_{ijp} \leq \left(M_p - t_{ij} - f_{jp} \right) x_{ijp} \quad \forall \ \ (i,j) \in A_p, \ \forall \ p \in \mathcal{P} \tag{4.13c}$$

The parameters M_p and f_{jp} are as described in Section 4.3.1.2. The implications of the triangle inequality (4.11) on the definitions of f_{jp} also carry over from the discussion there.

Constraints (4.13a) represent commodity-flow balances and enforce that the arrival time at node j equals the arrival time at its predecessor node plus the time it takes to travel from that predecessor to node j. Constraints (4.13b) simply require the vehicle to start from the depot at time 0 in each period. These constraints lead the commodity variables y_{ijp} to attain the value of the arrival time at node i, whenever $(i,j) \in A_p$ is part of the tour in period p, i.e., whenever $x_{ijp} = 1$. At the same time, constraints (4.13c) will ensure that $y_{ijp} = 0$, whenever $x_{ijp} = 0$.

The commodity-flow constraints (4.13) suffice to eliminate subtours.[8] Furthermore, the presence of commodity variables y_{ijp} enables us to explicitly enforce arrival-time consistency via constraints (4.14), achieving the arrival-time consistency requirement (4.7).

$$\sum_{j \in N_p^+(i)} y_{ijp} - \sum_{j \in N_q^+(i)} y_{ijq} \leq L \quad \forall \, i \in V_p \cap V_q, \ \forall \, (p,q) \in \mathcal{P} \times \mathcal{P} : p \neq q \tag{4.14}$$

Therefore, the model consisting of Eqs. (4.2)–(4.5), (4.13) and (4.14) is a complete and valid ConTSP formulation.

Finally, we remark that an alternative formulation results by utilizing commodity-flow constraints merely to eliminate subtours while using the IPEC (4.9), instead of constraints (4.14), to enforce arrival-time consistency. In such a case, one could apply projection techniques as in [139] to obtain a formulation in the space of the binary arc variables x_{ijp} only. Conversely, we also note that it is possible to model the ConTSP without introducing binary arc variables at all. This technique, which was used in [175] to model the TSPTW, utilizes a two-commodity flow representation of the cumulative travel time that is subsequently exploited in a branch-and-bound solution framework. The lower bound given by the linear programming relaxation of the two-commodity flow formulation would be identical to that of the single-commodity flow formulation [103].

8 The remark of the previous footnote applies also for the case of commodity-flow constraints.

4.3.1.4 *Sizes and Strength of Proposed Formulations*

Formulation 1 utilizes only the binary variables x_{ijp}, but features a factorially-large set of constraints. In contrast, Formulations 2 and 3 feature a polynomial number of constraints, but utilize additional variables. Table 4.1 provides a synopsis of the sizes of the three formulations.

Table 4.1: Sizes of ConTSP formulations.

Formulation	# of Variables		# of Constraints
	Binary	Continuous	
1	$\mathcal{O}(n^2h)$	—	$\mathcal{O}(n!^2h^2)$
2	$\mathcal{O}(n^2h)$	$\mathcal{O}(nh)$	$\mathcal{O}(n^2h + nh^2)$
3	$\mathcal{O}(n^2h)$	$\mathcal{O}(n^2h)$	$\mathcal{O}(n^2h + nh^2)$

It should be noted that, in principle, one can postulate a formulation for the ConTSP based on utilizing any valid asymmetric TSP formulation to model constraints (4.6), and we refer the interested reader to [130, 206, 222] for recent surveys of such formulations. As long as a formulation utilizes binary arc variables, one can in conjunction use Eqs. (4.9) to enforce the arrival-time consistency requirement (4.7) for a comprehensive ConTSP model. Forthermore, known tightness results and relationships between these asymmetric TSP formulations will persist in the case of their ConTSP counterparts. An experimental comparison of exact algorithms for the asymmetric TSP [222] reveals that the branch-and-cut algorithm of [114] that is based on the formulation introduced in [96] is computationally most efficient. We note that Formulation 1 utilizes the latter to achieve requirement (4.6) and enforces the arrival-time consistency requirement (4.7) via Eqs. (4.9). ConTSP formulations based on the other asymmetric TSP formulations are likely to reflect the same relative performance and consequently, we do not consider them in this study.

Conversely, Formulations 2 and 3 utilize travel time information while enforcing requirement (4.6) and by doing so, they enforce the arrival-time consistency requirement (4.7) via constraints (4.12) and (4.14) respectively, without having to introduce the IPEC (4.9). While this precludes a straightforward analysis of tightness relationships among the three ConTSP formulations we consider in this study, we present in Section 4.5.2 empirical evidence indicating that the dual bound obtained using the LP relaxation of Formulation 1 is always stronger than the bounds obtained using the LP relaxations of Formulations 2 and 3, while no empirical dominance relationship can be inferred between the latter two formulations.

4.3.2 *Valid Inequalities*

Let us define the ConTSP polytope, P_{CONTSP}, to be the convex hull of all integer feasible solutions of Formulation 1,

$$P_{\text{CONTSP}} = conv\left\{x \in \mathbb{R}^{\sum_{p\in\mathcal{P}}|A_p|} : x \text{ satisfies } (4.3), (4.4), (4.5), (4.8), (4.9)\right\}.$$

Several families of inequalities are valid for P_{CONTSP}. These include all inequalities that are valid for the asymmetric TSP. Note that an inequality that is valid for any of the asymmetric TSP instance associated with a period $p \in \mathcal{P}$ can be directly applied on the ConTSP instance. In our study, we consider Subtour Elimination and 2-matching Constraints. In addition, we consider the cross-period Inconsistent Path Elimination Constraints introduced earlier in Section 4.3.1.1. As discussed, the inclusion of SEC and IPEC is necessary for Formulation 1, which relies on these inequalities to eliminate subtours and enforce arrival-time consistency, respectively. The inequalities are redundant for Formulations 2 and 3, as long as the integrality restrictions (4.3) on the variables x_{ijp} are retained; however, they are still capable of strengthening the linear relaxation and should, thus, be used in conjunction with these formulations as well. In fact, the use of these inequalities as cutting planes is of fundamental importance from a practical point of view. For the instances we considered in our computational study (see Section 4.5), adding the cutting planes was very helpful in expediting the proof of optimality.

4.3.2.1 *Subtour Elimination Constraints*

These inequalities, which were introduced as constraints (4.8), forbid the occurrence of subtours and enforce the overall connectivity of the tour. Note however that, because of the degree constraints (4.4) and (4.5), the SEC defined by a vertex set $S \subset V_p \cup \{0\}$ and its complement $S^c = \left(V_p \cup \{0\}\right) \setminus S$ (see Eq. 4.15 below) are equivalent. To that end, one may use form (4.15) as an alternate to form (4.8). Numerical criteria, such as constraint sparsity (which depends on the size of set S), can be used to decide which of the two forms to utilize as a cutting plane in each case.

$$\sum_{(i,j)\in A(S^c)} x_{ijp} \leq |S^c| - 1 \quad \forall\, S \subseteq V_p \cup \{0\} : 2 \leq |S| \leq |V_p| - 1,\ \forall\, p \in \mathcal{P} \tag{4.15}$$

4.3.2.2 *2-matching Constraints*

The 2-matching Constraints (also known as Blossom Inequalities) are particular cases of a more general class of inequalities that is referred to as Comb Inequalities [112]. For a given period $p \in \mathcal{P}$ and given vertex sets $H, T_1, T_2, \ldots, T_k \subset V_p$, where $k \geq 3$ and odd, satisfying (i) $|H \cap T_i| = 1$ for $i = 1, \ldots, k$, (ii) $|T_i \setminus H| = 1$ for $i = 1, \ldots, k$, (iii) $T_i \cap T_j = \emptyset$

for $1 \leq i < j \leq k$, the corresponding 2-matching Constraint (2MC) is presented in Eq. (4.16).

$$\sum_{(i,j)\in A(H)} x_{ijp} + \sum_{i=1}^{k} \sum_{(i,j)\in A(T_i)} x_{ijp} \leq |H| + \frac{k-1}{2} \tag{4.16}$$

Such constraints are obtained by adding the degree constraints (4.4) and (4.5) for all $i \in H$, adding the subtour elimination constraints (4.8) defined by the vertex sets $S = T_i$ for all $i \in \{1, \ldots, k\}$, dividing by 2, and rounding down the right hand side to the nearest integer. More general Comb Inequalities may be obtained by relaxing conditions (i) and (ii) above as follows: $|H \cap T_i| \geq 1$ and $|T_i \setminus H| \geq 1$ for $i = 1, \ldots, k$. Constraint (4.16) may be equivalently cast in the form (4.17). As before, constraint sparsity can be used in each case to decide whether a cutting plane should be represented in form (4.16) or form (4.17).

$$\sum_{(i,j)\in\delta(H)\setminus\left(\bigcup_{i=1}^{k} A(T_i)\right)} x_{ijp} - \sum_{i=1}^{k} \sum_{(i,j)\in A(T_i)} x_{ijp} \geq 1 - k, \tag{4.17}$$

4.3.2.3 *Inconsistent Path Elimination Constraints*

These inequalities, which were introduced in Section 4.3.1.1, are cross-period constraints that forbid pairs of paths belonging to different periods that are inconsistent to simultaneously appear in the solution. Given a pair of paths $\{P, Q\}$, where $P = (v_1, \ldots, v_s)^p$ and $Q = (w_1, \ldots, w_t)^q$, the basic form of the inequality that forbids their simultaneous occurrence is presented in Eq. (4.18).

$$\sum_{i=1}^{s-1} x_{v_i v_{i+1} p} + \sum_{i=1}^{t-1} x_{w_i w_{i+1} q} \leq s + t - 3 \tag{4.18}$$

However, it is possible to strengthen this basic form (see also [16] for how to strengthen the infeasible path elimination constraints they proposed in the context of the TSPTW). In particular, it can be strengthened into the so-called Tournament Constraint (4.19).

$$\sum_{i=1}^{s-1} \sum_{j=i+1}^{s} x_{v_i v_j p} + \sum_{i=1}^{t-1} \sum_{j=i+1}^{t} x_{w_i w_j q} \leq s + t - 3 \tag{4.19}$$

Furthermore, for a given inconsistent pair of paths $\{P, Q\}$ as described above, let the paths obtained by reversing paths P and Q be denoted as $P' = (v_s, \ldots, v_1)^p$ and $Q' = (w_t, \ldots, w_1)^q$, respectively. If all three pairs of paths $\{P, Q'\}$, $\{P', Q\}$ and $\{P', Q'\}$ are also inconsistent, then the symmetric inequality (4.20), which corresponds to a lifting of inequality (4.18), is also valid and can be used instead of the latter.

$$\sum_{i=1}^{s-1} \left(x_{v_i v_{i+1} p} + x_{v_{i+1} v_i p}\right) + \sum_{i=1}^{t-1} \left(x_{w_i w_{i+1} q} + x_{w_{i+1} w_i q}\right) \leq s + t - 3 \tag{4.20}$$

Finally, if $\{P_1, Q_1\}$ and $\{P_2, Q_2\}$ are two inconsistent pairs of paths in periods p and q, such that P_1 is contained in P_2 and Q_1 is contained in Q_2, then the IPEC defined by $\{P_2, Q_2\}$ is dominated by the one defined by $\{P_1, Q_1\}$.

4.3.2.4 *Polyhedral Analysis*

A polyhedral analysis to determine whether the inequalities considered above are facet-defining for P_{CONTSP} is typically a difficult task. For a fixed graph size, $|V_p|$, and for fixed L, small changes in the travel times t_{ij} can change the dimension of P_{CONTSP} or even make the entire instance infeasible. Although there may exist specific instances in which individual inequalities coincide with facets of P_{CONTSP}, in general none of the three families of inequalities induces facets of the polytope, even if the instance is feasible. This observation is interesting inasmuch it implies that known polyhedral results for asymmetric TSP do not carry over to the case of ConTSP. For example, all SEC are known to induce facets of the asymmetric TSP polytope for the case of $n \geq 4$ [142], while all 2MC are known to be facet-defining for the asymmetric TSP polytope for the case of $n \geq 6$ [112]. However, as Proposition 4.1 shows, this is not true for the ConTSP.

Proposition 4.1. *The Subtour Elimination, 2-matching and Inconsistent Path Elimination Constraints do not induce facets of P_{CONTSP}, in general.*

Proof. Consider a ConTSP instance with $n = 7$, $h = 2$, $V_1 = \{1,2,3,4,5,6\}$ and $V_2 = \{1,7\}$. The arrival-time consistency requirement applies only to node 1, since node 1 is the only customer node common to both periods 1 and 2. Assume that the travel times are symmetric and are as depicted in Figure 4.1. For a maximum allowable arrival-time differential of $L = 1$, it is straightforward to see that all feasible ConTSP solutions are of the form $\{\langle 0,1,i_2,\ldots,i_6\rangle^1, \langle 0,1,7\rangle^2\}$, where $i_2,\ldots,i_6$ is some permutation of nodes 2 through 6. In this case, one may verify that the rank of the set of feasible solutions is 20 and that 0 does not participate in the affine hull of this set.[9] Therefore, the affine rank of P_{CONTSP} is 20 and $\dim(P_{\text{CONTSP}}) = 19$.

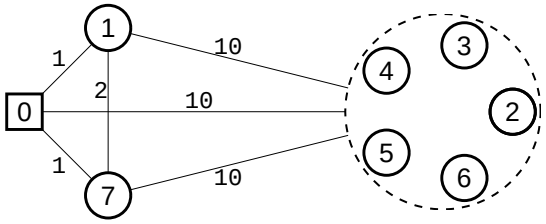


Figure 4.1: Travel times for the ConTSP instance considered in the proof of Proposition 4.1.

9 This may be verified numerically by showing the following linear system (in variables $\lambda_i \in \mathbb{R}$) to be infeasible: $\sum_{i\in F} x_i\lambda_i = 0$, $\sum_{i\in F} \lambda_i = 1$. Here, $\{x_i\}_{i\in F}$ is the set of feasible ConTSP solutions.

By a similar reasoning as above, it can be verified that the face induced by the SEC of the form (4.8), where $p = 1$ and $S = \{0, 2, 3\}$, is of dimension equal to 8 and, therefore, this inequality does not induce a facet of P_{CONTSP}. The face induced by the 2MC of the form (4.16), where $p = 1, k = 3, H = \{0, 1, 2\}$, $T_1 = \{0, 3\}$, $T_2 = \{1, 4\}$ and $T_3 = \{2, 5\}$, is of dimension equal to 7 and, therefore, this inequality does not induce a facet of P_{CONTSP}. In fact, it can be verified that none of the Comb Inequalities that are possible in period 1 (which include, as a special case, all 2-matching Constraints), induce facets of P_{CONTSP}. Finally, consider the IPEC of the form (4.9), where $P = (0, 2, 1)^1$ and $Q = (0, 7, 1)^2$. It can be verified that this inequality does not induce a face of P_{CONTSP}, i.e., there is no integral point in P_{CONTSP} that satisfies this inequality as an equality; therefore, it is not facet-defining for P_{CONTSP}. □

4.3.3 *Branch-and-cut Framework*

We implemented three separate branch-and-cut algorithms, one based on each of the formulations presented in Sections 4.3.1.1, 4.3.1.2 and 4.3.1.3. At the interest of working with sufficiently tractable linear programming (LP) relaxations, a subset of applicable constraints are initially ignored and added later as cutting planes, if necessary. More specifically, in the case of Formulation 1, the initial LP relaxation consisted only of the degree constraints (4.4) and (4.5), along with continuous bounds on the x variables (relaxation of the integrality restrictions (4.3)). In the case of Formulation 2, constraints (4.10)–(4.12) were considered, along with the degree constraints and variable bounds, while in the case of Formulation 3, the initial LP relaxation consisted of constraints (4.13) and (4.14), in addition to the degree constraints and variable bounds.

The SEC, 2MC and IPEC were considered in all cases as cutting planes and dynamically added back to the applicable model, when found to be violated, at each node of the search tree. At the root node, if we are unable to generate any additional violated inequalities, we permanently fix nonbasic x variables to their current values using reduced cost information. Let UB be the current (incumbent) upper bound (obtained through a heuristic or otherwise) and let LB be the global (root-node) lower bound obtained from the applicable LP relaxation. Let $\bar{c}_{ijp}$ be the reduced cost of each nonbasic variable x_{ijp} in the root-node LP solution. Then, if $x_{ijp} = 0$ and $LB + \bar{c}_{ijp} > UB$, we can fix this variable to zero. Conversely, if $x_{ijp} = 1$ and $LB - \bar{c}_{ijp} > UB$, then we can fix this variable to one. In a similar manner, using a local lower bound, one may set non-basic x variables to their current values in the sub-tree associated with each node of the search process. These fixings ensure that the variables are never branched upon in the corresponding sub-trees. The remainder of this section elaborates on our separation routines and associated protocols.

4.3.3.1 *Separation of Subtour Elimination Constraints*

In our implementation, we utilize both heuristic procedures and an exact scheme to separate violated SEC. In particular, we first attempt to separate these cuts using the separation heuristics described in [13, Chapter 6]. The advantage of these routines is that they often lead to the identification of several violated inequalities, as opposed to the one most violated inequality, and we have found it beneficial to be adding all such identified inequalities in a single cutting-plane iteration. When the heuristic routines do not identify any violations, we employ the exact separation scheme that was introduced in [208]. This is a polynomial-time routine that is based on a minimum-cut algorithm. The overall running time of our separation routine is dominated by the maximum-flow computations in the exact scheme, for which we use the Boost graph library [231].

4.3.3.2 *Separation of 2-matching Constraints*

In our implementation, we utilize both heuristic procedures and an exact scheme to separate violated 2MC. More specifically, we first attempt to separate these cuts using the so-called *odd-component*, *Grötschel-Holland* and *block* heuristic routines described in Applegate et al. [13, chap. 7]. Note that the latter heuristic also identifies violated general Comb Inequalities. As before, each of these heuristics may identify several violated inequalities, and we have found it beneficial to be adding all of them in a single cutting-plane iteration. If none of the heuristic routines succeeds in generating a violated cut, we employ the polynomial-time exact routine proposed in [186]. The running time of this routine is dominated by the constructor of the Gomory-Hu cut-tree and, in our implementation, we utilize the code in [232] for this purpose. Given its computational burden, we only employ the exact routine during cutting-plane iterations at the root node. Alternatively, one may use the polynomial-time exact algorithm proposed in [209] to separate violated 2MC.

4.3.3.3 *Separation of Inconsistent Path Elimination Constraints*

Our separation routines to identify violated IPEC is guided by Lemma 4.1. We use the following polynomial-time enumerative procedure to identify inconsistent pairs of paths. Given a (fractional) solution x^*, consider the support graph in each time period p. This support graph has the same node set as G_p and involves those arcs $(i,j) \in A_p$ for which $x^*_{ijp} > 0$. By considering every node in each period's support graph as a start node, elementary paths are grown in a depth-first fashion by moving along the incident outgoing arcs of strictly positive flow. Each path $P = (v_1, \ldots, v_s)^p$ is extended as long as the following two conditions hold: (i) the total flow on the path, $\sum_{i=1}^{s-1} x^*_{v_i v_{i+1} p}$, is strictly greater than $|P| - \frac{1}{2} = (s-1) - \frac{1}{2}$, and (ii) $v_s \neq 0$, i.e., the path has not reached the depot. Because of the first condition, an arc $(i,j) \in A_p$ is added to the path only if $x^*_{ijp} > 0.5$, while the path-growing procedure stops as soon as all incident outgoing arcs have weight

strictly less than 0.5. The degree constraint $\sum_{j \in N_p^+(i)} x^*_{ijp} = 1$ ensures that, for each node i, there is at most one incident outgoing arc with $x^*_{ijp} > 0.5$. Hence, there is at most one (unique) path extending out of each node and the path-growing procedure terminates in polynomial time.

After all such paths have been identified in all time periods, violated inequalities (4.18) are identified as follows: for each pair of nodes $v_1, v_s \in \left(V_p \cup \{0\}\right) \cap \left(V_q \cup \{0\}\right)$, we consider all pairs of paths in periods p and q such that each path contains both v_1 and v_s. If any of the conditions (i)–(iii) in Lemma 4.1 are satisfied for the pair of subpaths corresponding to v_1 and v_s as terminal nodes, then we have identified a violated inequality.

For every pair of paths that satisfy either condition (ii) or (iii) in Lemma 4.1, we check whether the corresponding IPEC of the form (4.18) can be lifted to the symmetric inequality (4.20) (see Section 4.3.2.3). If such a lifting is not possible, we add the corresponding Tournament Constraint (Eq. 4.19), because any pair of paths which violates inequality (4.18) also violates (4.19), and we have found it computationally beneficial to enforce the IPEC via the latter, stronger form. We remark that, if the solution x^* is integral and does not contain any subtours, then condition (i) of Lemma 4.1 is sufficient to guarantee the exactness of the separation procedure.

Our computational experiments indicated that the number of violated IPEC identified during a given cutting-plane iteration at a node of the branch-and-cut tree may be very large, especially for Formulation 1. For this reason, we attempt to reduce the number of constraints added to the constraint matrix in the following two ways: (i) we remove any dominated inequalities (see Section 4.3.2.3), and (ii) whenever the number of violated inequalities exceeds $2|V|$, we calculate for each cut (considered here as being of form 4.19) the ratio between its absolute violation at the current solution and the number of non-zero coefficients in the cut, namely

$$\frac{\sum_{i=1}^{s-1} \sum_{j=i+1}^{s} x^*_{v_i v_j p} + \sum_{i=1}^{t-1} \sum_{j=i+1}^{t} x^*_{w_i w_j q} - (s+t-3)}{\frac{1}{2}s\,(s-1) + \frac{1}{2}t\,(t-1)},$$

and we only add to the constraint matrix the $2|V|$ cuts with the highest such ratio.

4.3.3.4 *Separation Protocol*

We attempt to separate violated inequalities at each node of the branch-and-cut tree in the following sequence: (i) SEC for each time period $p \in \mathcal{P}$, (ii) 2MC for each time period $p \in \mathcal{P}$, and (iii) IPEC for each pair of time periods $p, q \in \mathcal{P} : p < q$. When we identify violated members of any of the above families of cuts in a given time period, then no attempt is made to separate members of other families of cuts that appear further in the separation sequence and that involve variables from that time period. At any given node other than the root node, whenever new cuts are added but the objective function

value did not improve by at least 0.1% in the last 20 cutting-plane iterations, we exit the separation sequence and resort to branching. At the root node, we have found it beneficial to keep separating as many cuts as possible before branching, even if the lower bound does not show significant improvement at that moment. Finally, we remark that, since all inequalities considered are valid globally (i.e., throughout the branch-and-cut tree), any violated inequality identified is added as a global cut.

4.4 DECOMPOSITION ALGORITHM

In this section, we describe a scheme to decompose the ConTSP into single period traveling salesman problems with arrival-time windows, and a branch-and-bound search approach that converges to the optimal ConTSP solution. The algorithm uses the following result.

Lemma 4.2. *Given a feasible ConTSP solution* $\{T_1, \ldots, T_{|\mathcal{P}|}\}$*, there exists a feasible solution to the linear system* (4.1) *that satisfies the following disjunction*

$$\left[a_i^p \geq \beta - \frac{L}{2} \quad \forall\, p \in \mathcal{P}_i\right] \vee \left[a_i^p \leq \beta + \frac{L}{2} \quad \forall\, p \in \mathcal{P}_i\right] \quad \forall\, \beta \in \mathbb{R},\, \forall\, i \in V_c \tag{4.21}$$

Proof. Let us assume that no feasible solution to the linear system (4.1) satisfies the above disjunction, i.e., every such solution satisfies $a_i^q < \beta - (L/2)$ and $a_i^r > \beta + (L/2)$ for some $i \in V_c$, $(r, q) \in \{\mathcal{P}_i \times \mathcal{P}_i : r \neq q\}$ and $\beta \in \mathbb{R}$. As a result, $a_i^r - a_i^q > L$ and, since $\Delta a_i^{\max} \geq a_i^r - a_i^q$ (by definition), it follows that $\Delta a_i^{\max} > L$. However, this is a contradiction, since the given ConTSP solution is feasible and, by definition, it satisfies the requirement that there exists a feasible solution to (4.1) for which the maximum arrival-time differential is no greater than the maximum allowable value: $\Delta a_i^{\max} \leq \max_{i \in V_c} \Delta a_i^{\max} \leq L$. □

4.4.1 *Outline of the Decomposition Algorithm*

The exact algorithm is based on a branch-and-bound search that uses the above disjunction to create valid branches whenever the candidate solution is infeasible with respect to arrival-time consistency. As a consequence, each node of the search tree corresponds to a set of TSPTW instances (one for each period), which can be solved separably so as to process the node. Note that each node can be characterized by a vector of applicable time windows (one for each customer), which we denote as $\mathcal{TW}$.

1. *Initialize.* Let time windows $\left[e_i^0, \ell_i^0\right] := [0, D]$, $\forall\, i \in V_c$, and set root node $\mathcal{TW}^0 \leftarrow \left(\left[e_1^0, \ell_1^0\right], \ldots, \left[e_n^0, \ell_n^0\right]\right)$, node queue $\mathcal{N} \leftarrow \left\{\mathcal{TW}^0\right\}$, upper bound $z_{\text{ub}} \leftarrow +\infty$ and optimal solution $T^* \leftarrow \emptyset$.
2. *Check convergence.* If $\mathcal{N} = \emptyset$, then stop. T^* is the optimal solution with cost z_{ub}.

3. *Select node.* Select a node $\mathcal{TW}$ from $\mathcal{N}$ using a best-bound strategy, and set $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\mathcal{TW}\}$.
4. *Process node.* For each $p \in \mathcal{P}$:
   a) Solve the TSPTW on graph G_p with time windows $\mathcal{TW}_i$ for each $i \in V_p$.
   b) If the TSPTW instance is infeasible, then go to step 2.
   Otherwise, let $T_p = \left\langle v_{p,0} = 0, v_{p,1}, \ldots, v_{p,|V_p|}, 0 \right\rangle$ be its optimal tour.

   Define $T := \left\{T_1, \ldots, T_{|\mathcal{P}|}\right\}$ as the resulting ConTSP solution.
5. *Fathom by bound.* If $\sum_{p\in\mathcal{P}} c(T_p) \geq z_{\text{ub}}$, then go to step 2.
6. *Check feasibility.* Let $(\hat{d}, \hat{a})$ be the optimal solution of the following, feasible linear program:

$$\min_{a,d} \; d \tag{4.22a}$$

$$\text{s.t. } d \geq a_i^p - a_i^q \qquad \forall\, i \in V_p \cap V_q, \forall\, (p,q) \in \mathcal{P} \times \mathcal{P} : p \neq q \tag{4.22b}$$

$$a^p_{v_{p,0}} = 0 \qquad \forall\, p \in \mathcal{P} \tag{4.22c}$$

$$a^p_{v_{p,k}} \geq a^p_{v_{p,k-1}} + t_{v_{p,k-1}v_{p,k}} \qquad \forall\, k \in \{1, \ldots, |V_p|\}, \forall\, p \in \mathcal{P} \tag{4.22d}$$

$$a^p_{v_{p,|V_p|}} + t_{v_{p,|V_p|}0} \leq D \qquad \forall\, p \in \mathcal{P} \tag{4.22e}$$

$$a_i^p \in \mathcal{TW}_i \qquad \forall\, i \in V_p, \forall\, p \in \mathcal{P} \tag{4.22f}$$

$$d \in \mathbb{R}_+ \tag{4.22g}$$

   If $\hat{d} \leq L$, then T constitutes a feasible ConTSP solution; set $T^* \leftarrow T$ and $z_{\text{ub}} \leftarrow \sum_{p\in\mathcal{P}} c(T_p)$, and go to step 2.
7. *Select branching rule.* Let $i^* \in V_c$ be the customer for which $\Delta\hat{a}^{\max}_{i^*} = \hat{d}$ (arbitrarily breaking ties, if necessary), and let $\underline{a} = \min_{p\in\mathcal{P}_{i^*}} \hat{a}^p_{i^*}$, $\bar{a} = \max_{p\in\mathcal{P}_{i^*}} \hat{a}^p_{i^*}$ and $\beta^* = (\underline{a} + \bar{a})/2$.
8. *Branch.* Instantiate two children nodes from the parent node, and tighten the time window for i^*, as follows:
   - $\mathcal{TW}^{\text{L}} \leftarrow \mathcal{TW}$, $\ell^{\text{L}}_{i*} = \beta^* + L/2$
   - $\mathcal{TW}^{\text{R}} \leftarrow \mathcal{TW}$, $e^{\text{R}}_{i*} = \beta^* - L/2$

   Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{\mathcal{TW}^{\text{L}}, \mathcal{TW}^{\text{R}}\}$, and go to step 3.

Note that the chosen (i^*, β^*) pair defines a valid branching rule because (i) it is valid as per Lemma 4.2, and (ii) the solution T obtained in step 4 violates the disjunction (4.21) for customer i^*. More specifically, since $\bar{a} - \underline{a} = \Delta\hat{a}^{\max}_{i^*} = \hat{d} > L$, it follows that $\underline{a} < \beta^* - L/2$ and $\bar{a} > \beta^* + L/2$. Also note that the above algorithm converges in finite time because (i) the union of the feasible regions of the children nodes, $\mathcal{TW}^{\text{L}}$ and $\mathcal{TW}^{\text{R}}$, created in step 8, is strictly smaller than that of the parent node, $\mathcal{TW}$, (ii) there are only a finite number of single-period Hamiltonian cycles to consider, and (iii) every TSPTW subproblem can be

solved in finite time. In practice, of course, the actual number of TSPTW subproblems that have to be solved is fairly small (see Section 4.4.3).

It is also interesting to point out that the lower bounds at each node, which are obtained by solving separably in step 4 a set of single-period TSPTW instances (with or without waiting), can be interpreted as Lagrangean bounds obtained by dualizing the cross-period arrival-time consistency constraints in an integer programming formulation of the ConTSP with zero multiplier weights. While it is possible to update these multipliers and approach the Lagrangean dual bound, our preliminary computational experiments utilizing the subgradient and cutting plane methods to perform such updates suggested that this does not pay off. The solution of the subproblems (TSPTW) at each node became more difficult at the benefit of a relatively minor improvement in the final lower bounds. Finally, we highlight that, if waiting is not allowed, $(\hat{d}, \hat{a})$ in step 6 can be computed in closed form, instead of solving a linear program, since in this case constraint (4.22d) becomes an equality.

4.4.2 *An Illustrative Example*

Consider a ConTSP instance with $n = 5$, $|\mathcal{P}| = 2$, $V_1 = \{1, 2, 3, 4\}$, $V_2 = \{2, 3, 5\}$, $D = 16$ and $L = 2$. The arrival-time consistency requirement applies to nodes 2 and 3 only, since they are the only customer nodes common to both periods 1 and 2. Assume that the costs and travel times coincide, are symmetric and are as depicted in Figure 4.2. For simplicity of exposition, assume also that waiting is not allowed. Figure 4.3 presents the branch-and-bound tree of the algorithm for this example. Each figure represents the solution at a given node of the branch-and-bound tree, where the sequence of square boxes in each time period represents the actual tour, their positions along the time axis represent the arrival-times, while the sum (across the two time periods) of the arrival times at the final depot (rightmost "0" boxes) represents the objective value of the corresponding solution. Whenever a node is infeasible, the offending boxes are depicted in gray color, and the corresponding branching decisions (if applicable) are indicated along the branching arrows.

Node A constitutes the root node where no time windows are active. Since customer 3 violates the arrival-time consistency requirement, $\Delta a_3^{\max} = a_3^2 - a_3^1 = 3 > 2 = L$, node A is branched upon to obtain nodes B and C with time windows $a_3^p \in [0, 9.5], \forall\ p \in \mathcal{P}$, and $a_3^p \in [7.5, 16], \forall\ p \in \mathcal{P}$, respectively. Node C represents a feasible solution and it provides the first incumbent upper bound, $z_{\text{ub}} = 29$. By a similar reasoning as above, branching on node B results into nodes D and E, while branching on node E results in nodes F and G. Nodes D and F are infeasible because they do not respect the route duration limit $D = 16$ in period 2, and are thus fathomed (step 4b).[10] Node G represents another

10 The infeasible routes depicted in Figure 4.2, nodes D and F, period 2, are provided for reference; they correspond to routes with minimum total travel time that adhere to the imposed time windows but without regard to the route duration limit.

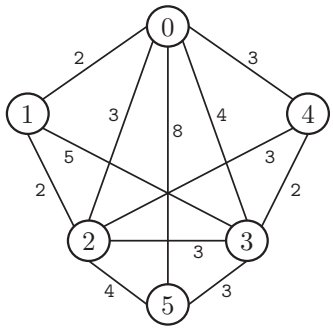


Figure 4.2: Travel times and costs for the illustrative example.

feasible ConTSP solution, which happens to attain an equal objective value as the current incumbent obtained at node C. Node G is technically fathomed by bound (step 5).

4.4.3 *Algorithmic Details: Solving TSPTW Instances*

Step 4a of the main algorithm presented in Section 4.4.1 involves the repeated solution of TSPTW subproblems, which provide us with the lower bounds necessary to drive the overall branch-and-bound search. In fact, after branching has occurred in step 8 of the algorithm, at most $|\mathcal{P}|$ (out of a total of $2|\mathcal{P}|$) TSPTW subproblems have to be solved across the two sibling nodes, while the remaining TSPTW solutions can be directly transferred from parent subproblems. This is true because the optimal TSPTW tour in a given time period cannot simultaneously violate both terms of the branching disjunction (4.21) (although it may still satisfy both terms simultaneously). Therefore, as far as a given time period is concerned, a TSPTW needs to be solved only at most once across the two children nodes. In addition, it can be shown that at least one TSPTW subproblem (or at least two, when waiting is not permitted) will have to be solved across the two siblings. Recall that the time windows in the TSPTW instances of the children nodes differ from those of the parent in exactly one component, namely their i^* component (and that too only in either their latest or earliest arrival times, but not both). In particular $\mathcal{TW}^{\mathrm{L}}_{i^*}$ has a tighter (i.e., smaller) latest arrival time, giving rise to at least one TSPTW instance for which the corresponding parent tour is not feasible, while $\mathcal{TW}^{\mathrm{R}}_{i^*}$ has a tighter (i.e., larger) earliest arrival time, which (when waiting is not permitted) gives rise to at least one more TSPTW instance for which the corresponding parent tour is not feasible.

In order to illustrate the above properties, let us focus on the search tree depicted in Figure 4.3. We observe that, in this two-period example, exactly 2 TSPTW subproblems need to be solved per each pair of sibling nodes. More specifically, the optimal tour of period 1 in node A remains optimal in its child node B even after imposing the time window $a_3 \in [0, 9.5]$, while the optimal tour of period 2 in node A remains optimal in node C even after imposing the time window $a_3 \in [7.5, 16]$. Hence, only node B's

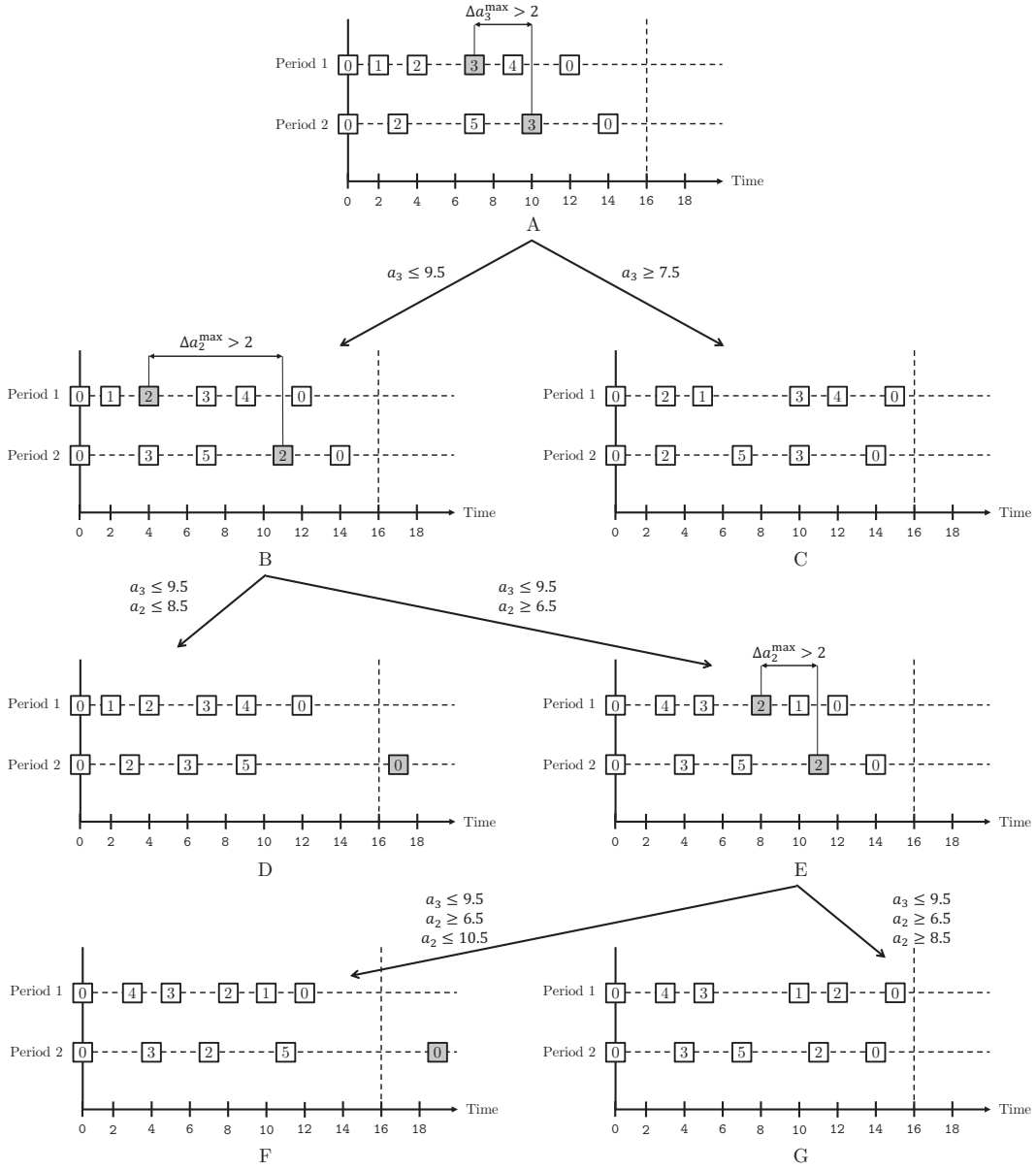


Figure 4.3: The search tree of our algorithm for the illustrative example.

period 2 and node C's period 1 TSPTW subproblems need to be solved at the next tree level. Similar behavior can be observed after branching upon nodes B and E as well.

Below we review the relevant literature on the exact solution of the TSPTW, and we discuss the approach we used as part of our implementation. We also discuss a heuristic procedure for generating valid initial upper bounds, which can be used to warm-start the TSPTW solution process.

4.4.3.1 *Literature Review.*

Branch-and-bound schemes to solve variants of the TSPTW have been presented in a number of papers. Christofides, Mingozzi, and Toth [83] derived lower bounds using a state-space relaxation approach, wherein the state space associated with a given dynamic programming (DP) recursion is relaxed, Baker [23] derived lower bounds from the solution of a time-constrained longest path problem, and Langevin et al. [175] used a two-commodity flow formulation to obtain lower bounds. The first two papers considered problems that seek to minimize the total makespan, while the third paper also considered the traditional objective of minimizing travel cost. In addition, many researchers have proposed DP-based algorithms for the TSPTW. Dumas et al. [108] developed advanced pre-processing routines and elimination tests to reduce the number of states and state transitions within a DP approach. Mingozzi, Bianco, and Ricciardelli [199] and Balas and Simonetti [26] presented algorithms for the TSPTW with additional precedence restrictions. Li [187] reported the solution of instances with up to 233 vertices using a bi-directional and resource-bounded label-correcting algorithm. Baldacci, Mingozzi, and Roberti [31] presented new state space relaxations for the TSPTW to compute tight lower bounds and solved all but one instance from existing TSPTW benchmark sets to optimality. More recently, Tilk and Irnich [252] generalized the ideas of Baldacci, Mingozzi, and Roberti [31] for the case when the objective is to minimize the makespan or the total duration time. Furthermore, approaches based on constraint programming have also been developed for the TSPTW. Pesant et al. [212] used redundant constraints to improve a constraint-programming model of the TSPTW, while Focacci, Lodi, and Milano [117] proposed a hybrid approach that merges constraint programming ideas with more traditional optimization techniques.

Branch-and-cut algorithms for the TSPTW have been developed in [17, 98]. Ascheuer, Fischetti, and Grötschel [17] considered several MILP formulations for the asymmetric version of the problem and computationally compared them within a branch-and-cut approach. They used several families of cutting planes in their scheme, along with problem-specific local search heuristics and pre-processing routines. Dash et al. [98] presented a new formulation for the TSPTW based on partitioning the time windows into sub-windows called buckets. They used several families of cutting planes derived from their bucket formulation, as well as those developed in [17]. Recently, Kara et al. [165] presented a new two-commodity flow formulation for the TSPTW with the objective

of minimizing total duration time and solved their resulting model with a commercial MILP solver.

All of the previous papers assume that the vehicle can wait as long as needed at any customer location. For the ConTSP without waiting, our solution scheme involves the repeated solution of TSPTW instances where waiting at customer locations is not allowed. Consequently, we cannot directly use these approaches. In particular, extending the state-of-the-art DP algorithm of Baldacci, Mingozzi, and Roberti [31] is not particularly straightforward when there are constraints prohibiting waiting, as the latter would invalidate many of the key dominance rules used by the authors. In our implementation, we use the branch-and-cut algorithm of Ascheuer, Fischetti, and Grötschel [17] to solve the TSPTW in step 4 of the main algorithm described in Section 4.4. In the following, we describe the MILP models and the branch-and-cut approach as well as appropriate modifications that must be made in order to handle the case of no waiting.

For ease of notation, we shall drop the subscript p and use V_c to denote the set of customers requiring service in time period $p \in \mathcal{P}$, with A being the corresponding arc set. We shall assume that $[e_i, \ell_i]$ is the time window associated with customer $i \in V_c$. For each node $i \in V_c \cup \{0\}$, we let N_i^+ denote the set of nodes j for which there is an arc from i to j; that is, $N_i^+ := \{j \in V : (i,j) \in A\}$. Similarly, let $N_i^- := \{j \in V : (j,i) \in A\}$. Finally, given a subset of nodes $S \subseteq V_c \cup \{0\}$, let $A(S)$ be the set of arcs with both end points in S; that is, $A(S) := \{(i,j) \in S \times S : i \neq j\}$.

4.4.3.2 *MILP Model 1.*

This model extends the polyhedral formulation for the TSP introduced in [96]. Let $x_{ij} \in \{0,1\}$ encode whether arc (i,j) is part of the optimal tour. The following MILP model is then valid for the TSPTW:

$$\min_x \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{4.23a}$$

$$\text{s.t.} \quad x_{ij} \in \{0,1\} \qquad \forall\ (i,j) \in A \tag{4.23b}$$

$$\sum_{j\in N_i^+} x_{ij} = \sum_{j\in N_i^-} x_{ji} = 1 \qquad \forall\, i \in V_c \cup \{0\} \tag{4.23c}$$

$$\sum_{(i,j)\in A(S)} x_{ij} \leq |S| - 1 \qquad \forall\, S \subseteq V_c \cup \{0\} : |S| \in \{2, \cdots, |V_c| - 1\} \tag{4.23d}$$

$$\sum_{(i,j)\in P} x_{ij} \leq |P| - 1 \qquad \forall \text{ infeasible paths } P \tag{4.23e}$$

Inequalities (4.23e) are Infeasible Path Elimination Constraints and they forbid the occurrence of paths whose existence would render the resulting TSPTW solution infeasible. Let $P = (v_1, \ldots, v_s)$, with $s > 1$ and all v_k nodes unique, denote a path that is formed by the arcs in the set $\{(v_k, v_{k+1}) : k = 1, \ldots, s-1\}$, where $(v_k, v_{k+1}) \in A$ for all $k = 1, \ldots, s-1$.

Let us denote the earliest and latest possible arrival times at node v_k by epat_{v_k} and lpat_{v_k}. These quantities can be computed as follows:

$$\text{epat}_{v_1} := e_{v_1}$$
$$\text{lpat}_{v_1} := \ell_{v_1}$$
$$\text{epat}_{v_k} := \begin{cases} \max\left\{e_{v_k}, \text{epat}_{v_{k-1}} + t_{v_{k-1},v_k}\right\} & \text{if waiting is allowed} \\ \text{epat}_{v_{k-1}} + t_{v_{k-1},v_k} & \text{if waiting is not allowed} \end{cases} \quad \text{for } k = 2,\dots,s$$
$$\text{lpat}_{v_k} := \begin{cases} \max\left\{e_{v_k}, \text{lpat}_{v_{k-1}} + t_{v_{k-1},v_k}\right\} & \text{if waiting is allowed} \\ \text{lpat}_{v_{k-1}} + t_{v_{k-1},v_k} & \text{if waiting is not allowed} \end{cases} \quad \text{for } k = 2,\dots,s$$

Note that, for the case when waiting is not allowed, these quantities depend only on the time window restrictions of the first node in the path (since the vehicle needs to keep moving at all times). Sufficient conditions for the infeasibility of a path P can then be given in the following lemma, which simply checks the arrival time at the path's last node.

Lemma 4.3. *A path $P = (v_1,\dots,v_s)$, is infeasible, if any of the following conditions holds:*

(i) P violates the latest arrival-time window of its last node, i.e., $\text{epat}_{v_s} > \ell_{v_s}$.

(ii) P violates the earliest arrival-time window of its last node, i.e., $\text{lpat}_{v_s} < e_{v_s}$.

Note that Lemma 4.3 augments the sufficient condition presented by Ascheuer, Fischetti, and Grötschel [17] to consider the case when waiting time is not allowed. More specifically, condition (ii) can only yield a violation in this situation.

4.4.3.3 *MILP Model 2.*

This TSPTW model corresponds to the single-commodity flow formulation that was presented in [17]. In this formulation, y_{ij} encodes the arrival time at node $i \in V$, if arc $(i,j) \in A$ is part of the tour, and 0 otherwise. Note that, if waiting is not allowed, then Constraint (4.24e) in the following model must be converted to an equality.

$$\min_{x,y} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{4.24a}$$
$$\text{s.t.} \quad x_{ij} \in \{0,1\},\ y_{ij} \in \mathbb{R}_+ \qquad \forall\ (i,j) \in A \tag{4.24b}$$
$$\sum_{j\in N_i^+} x_{ij} = \sum_{j\in N_i^-} x_{ji} = 1 \qquad \forall\, i \in V_c \cup \{0\} \tag{4.24c}$$
$$\sum_{k\in N_0^+} y_{0k} = 0 \tag{4.24d}$$
$$\sum_{k\in N_j^+} y_{jk} \geq \sum_{i\in N_j^-} \left(y_{ij} + t_{ij}x_{ij}\right) \qquad \forall\, j \in V_c \tag{4.24e}$$

$$e_i x_{ij} \leq y_{ij} \leq \ell_i x_{ij} \qquad \forall\ (i,j) \in A \tag{4.24f}$$

We remark that, since arrival times are explicitly encoded as decision variables, it is possible to extend this model so as to incorporate costs associated with vehicle idling. This is relevant in situations where the total costs depend on the amount of time the vehicle is in use (e.g., when drivers are paid according to their actual working time, or when driver overtime costs must be accounted).

4.4.3.4 *Branch-and-cut Approach.*

For the TSPTW subproblems that indeed warrant a new solution to be computed, we implemented two separate branch-and-cut algorithms, based on each of the formulations presented above. In the case of Model 1, the initial LP relaxation consisted only of the degree constraints (4.23c), along with continuous bounds on the x variables (relaxation of the integrality restrictions (4.23b)), while in the case of Model 2, constraints (4.24c)–(4.24f) were considered, in addition to the variable bounds. Note that before we call the exact routine, we modify the instance to obtain an equivalent one by tightening the time windows $[e_i, \ell_i]$ and reducing the arc set A using the pre-processing routines described in [17, 98]. While doing so, we identify precedence relationships among nodes; these precedences are used in the above pre-processing phase and also during the separation and addition of valid inequalities as cutting planes.

We considered the following inequalities as cutting planes [17], which we attempt to separate in the order they are listed. When we identify violated members of any of the below families of cuts, then no attempt is made to separate members of other families of cuts that appear further in the separation sequence.

1. Subtour elimination constraints [96]: We separated these inequalities using the routines and protocol described in Section 4.3.3.
2. Blossom inequalities [13]: We employ the same exact separation procedures as described in Section 4.3.3.
3. $\pi-$, $\sigma-$ and $(\pi,\sigma)-$ inequalities [25]: We use the heuristic separation procedures for the weak version of these inequalities described in [25].
4. Infeasible path elimination constraints [16]: We use these inequalities in their tournament form and separate them using the polynomial time path-growing scheme described in [17]. In doing so, we use the sufficient conditions described in Lemma 4.3 to check if a candidate path is infeasible.

4.4.3.5 *Protocol.*

Computational experiments [17] have indicated that none of the two branch-and-cut models dominates the other in terms of performance across the whole range of instances they had considered. In particular, the authors had observed that Model 2 performed

well on problems when only few time windows were active, whereas Model 1 performed best otherwise. We were able to confirm this observation on TSPTW instances arising in the context of our decomposition scheme. To that end, we have decided to employ both approaches and, in the remainder of this section, we shall describe our protocol in terms of how we utilized the two TSPTW solvers in the context of step 4a of our main algorithm.

Since most of the generated TSPTW instances have wide, overlapping time windows, the branch-and-cut algorithm based on Model 2 was more robust. Furthermore, pretests suggested that in the vast majority of cases, the algorithm based on Model 1 can either solve an instance very fast (on the order of a few seconds), or the solution time may easily exceed many minutes. We therefore decided to allocate a small computational budget for the branch-and-cut algorithm based on Model 1 in our solution protocol. In particular, we first attempt to solve the TSPTW instance using the branch-and-cut algorithm based on Model 1 with a time budget of ρ. If the algorithm terminated successfully, we continue on to step 4b. Otherwise, we attempt to solve the TSPTW instance using the branch-and-cut algorithm based on Model 2.

4.4.3.6 *Warm-starting.*

Providing the branch-and-cut solver with a valid initial upper bound on the cost of the optimal TSPTW tour can significantly speed up the search, as more parts of the search tree can be fathomed early in the process. Below we describe the heuristic procedure we used to generate such upper bounds. The procedure attempts to "repair" the optimal tour of the parent node and generate a tour that is feasible for the applicable child node as well.

Consider a parent node $\mathcal{TW}$ (already processed) with optimal solution $T = \{T_1, \ldots, T_{|\mathcal{P}|}\}$ as obtained from step 4 of the main algorithm. Consider also one of its child nodes $\mathcal{TW}^j$, where $j \in \{\mathrm{L}, \mathrm{R}\}$, as created after branching (based on customer i^*) in step 8 of main algorithm. Finally, consider a period $p \in \mathcal{P}_{i^*}$, such that T_p is not feasible for $\mathcal{TW}^j$. This means that the parent solution for period p cannot be directly transferred to the child, and hence, warm-starting is sought to aid the search towards a new solution. The below procedure computes a valid upper bound, ub_p^j, on the cost of the child's period–p optimal TSPTW tour.

1. *Initialize.* Set $ub_p^j \leftarrow z_{ub} - \sum_{q \in \mathcal{P}: q \neq p} c(T_q)$, where z_{ub} is the currently applicable upper bound on the optimal ConTSP solution.
2. Let $k \in \{1, \cdots, |V_p|\}$ be the position of customer i^* in tour T_p, and let index set K be as follows: $K = \{1, \ldots, k-1\}$, if $j = \mathrm{L}$, or $K = \{k+1, \ldots, |V_p|\}$, if $j = \mathrm{R}$.
    a) *Node reinsertion heuristic.* For each $l \in K$: Remove customer i^* from position k in the tour T_p and reinsert it at position l to obtain tour T'. If T' is feasible for $\mathcal{TW}^j$ and $c(T') < ub_p^j$, set $ub_p^j \leftarrow c(T')$.

b) *Forward arc reinsertion heuristic (only if $k \neq |V_p|$).* For each $l \in K \setminus \{|V_p|\}$: Remove customer i^* and its immediately following customer from positions k and $k+1$ in the tour T_p and reinsert them at positions l and $l+1$, respectively, to obtain tour T'. If T' is feasible for $\mathcal{TW}^j$ and $c(T') < ub_p^j$, set $ub_p^j \leftarrow c(T')$.

c) *Backward arc reinsertion heuristic (only if $k \neq 1$).* For each $l \in K \setminus \{1\}$: Remove customer i^* and its immediately preceding customer from positions k and $k-1$ in the tour T_p and reinsert them at positions l and $l-1$, respectively, to obtain tour T'. If T' is feasible for $\mathcal{TW}^j$ and $c(T') < ub_p^j$, set $ub_p^j \leftarrow c(T')$.

Note that solving the (up to $|\mathcal{P}|$) TSPTW subproblems can be conducted in parallel, if desired. In our implementation, we solved these subproblems serially on a single CPU thread, starting with the lowest indexed time period. In this setting, one may capitalize on subproblems that have already been solved so as to obtain improved upper bounds in step 1 of the above procedure as follows:

$$ub_p^j \leftarrow z_{ub} - \sum_{q \in \mathcal{P}: q < p} c(T_q^j) - \sum_{q \in \mathcal{P}: q > p} c(T_q),$$

where T_q^j is the recently computed optimal TSPTW tour in time period $q \in \mathcal{P}$ of the node under consideration ($\mathcal{TW}^j$).

4.4.4 *Algorithmic Details: Branching*

It should be noted that the choice of $\beta^* = (\underline{a} + \bar{a})/2$ in step 7 of the main algorithm (see Section 4.4.1) is not the only one that defines a valid branching rule. Indeed, any $\beta^* \in (\underline{a} + L/2, \bar{a} - L/2)$ will define a valid branching rule. In this regard, we tested a number of strategies, including the strategy where β^* is chosen such that the number of TSPTW instances that have to be solved across the two children nodes is minimized; that is, the value β^* that minimizes the function

$$V(\beta) = \min_{I \in \{0,1\}^{|\mathcal{P}_{i^*}|}} \left\{ \sum_{p \in \mathcal{P}_{i^*}} I_p : (\beta - L/2)(1 - I_p) \leq \hat{a}_{i^*}^p \leq (\beta + L/2) + D\, I_p,\ \forall\, p \in \mathcal{P}_{i^*} \right\}.$$

Here, I_p indicates whether a TSPTW instance would have to be solved in period p in either of the two children nodes, while D acts a big-M coefficient. For the instances we considered, no strategy seemed to consistently outperform the original branching rule, $\beta^* = (\underline{a} + \bar{a})/2$; hence, the latter was adopted for our computational experiments.

However, these preliminary investigations led to an interesting finding. Allowing waiting causes the linear program (4.22), which generates arrival time values $\hat{a}$ in step 6 of the algorithm, to often exhibit significant degeneracy. This happens because of route duration slackness, making it possible to shift forward in time the arrival at some (at least one) customer locations in all time periods, while still maintaining ConTSP feasibility

that features the same maximum arrival-time differential $\hat{d}$. Using this degeneracy to our advantage, we have found it beneficial to shift the arrival times to their earliest possible values by minimizing the cumulative waiting time, which can be achieved via the following two-step procedure. We first solve the linear program (4.22) to obtain the optimal $\hat{d}$ value. Assuming $\hat{d} > L$ (otherwise the value of $\hat{a}$ is irrelevant), we solve the auxiliary linear program (4.25) to obtain an equally-optimal, yet more promising solution $\hat{a}$.

$$\min_{a,w} \sum_{p\in\mathcal{P}} \sum_{i\in V_p} w_i^p \tag{4.25a}$$

$$\text{s.t. } \hat{d} \geq a_i^p - a_i^q \qquad \forall\, i \in V_p \cap V_q, \forall\, (p,q) \in \mathcal{P}\times\mathcal{P} : p \neq q \tag{4.25b}$$

$$a_{v_{p,0}}^p = 0 \qquad \forall\, p \in \mathcal{P} \tag{4.25c}$$

$$a_{v_{p,k}}^p = a_{v_{p,k-1}}^p + t_{v_{p,k-1}v_{p,k}} + w_{v_{p,k}}^p \qquad \forall\, k \in \{1,\ldots,|V_p|\}, \forall\, p \in \mathcal{P} \tag{4.25d}$$

$$a_{v_{p,|V_p|}}^p + t_{v_{p,|V_p|}0} \leq D \qquad \forall\, p \in \mathcal{P} \tag{4.25e}$$

$$a_i^p \in \mathcal{TW}_i \qquad \forall\, i \in V_p, \forall\, p \in \mathcal{P} \tag{4.25f}$$

$$w_i^p \in \mathbb{R}_+ \qquad \forall\, i \in V_p, \forall\, p \in \mathcal{P} \tag{4.25g}$$

4.4.5 *Algorithmic Details: Stalling*

Computational experiments using a first implementation of our algorithm revealed that, for many ConTSP instances we considered, the majority of the computational time was spent in processing those nodes that entailed relatively less tractable TSPTW subproblems. Our experience showed that, on average, about 80% of the time was spent in the solution of fewer than 10% of the TSPTW subproblems encountered in the search. Moreover, the geometric average (across all of the ConTSP benchmarks) of the ratio of the maximum to median time to solve subproblems was in excess of 700. It can thus be argued that detecting and bypassing–whenever possible–these time-consuming parts would significantly reduce the overall running time of our algorithm.

We consider the solution process of a TSPTW instance to have stalled if it has not terminated successfully within a CPU time of τ, and we recognize that changing the actual values we imposed as time windows has the potential to significantly change the CPU time required to solve it. In this context, a different set of values for the time windows can be obtained by applying the disjunction (4.21) for a different (i,β) pair than the one originally used for branching the parent node, assuming that another valid such pair exists. To that end, whenever we encounter stalling, we attempt to create a new set of TSPTW instances by backtracking to the parent node and considering an alternative branching decision. In order to implement this, we store the list of alternative branching decisions, $\mathcal{IB}$, as part of a node's characteristic data (along with $\mathcal{TW}$) at the time of node creation (branching step). We also modify steps 4a and 7 of the original algorithm as follows:

4a′ If $\mathcal{IB} \neq \emptyset$, let $t^{\text{lim}} = \tau$; else, let $t^{\text{lim}} = \infty$. Solve the TSPTW on graph G_p with time windows $\mathcal{TW}_i$ for each $i \in V_p$, specifying a CPU time limit of t^{lim}. If this time limit is reached, we backtrack as follows: (i) we remove from the search tree the current node's sibling or (if already processed) all its descendants still remaining in the tree, (ii) we reset the time windows of the current node to be the same as its parent's, (iii) we select (i^*, β^*) to be the first element of $\mathcal{IB}$, and we set $\mathcal{IB} \leftarrow \mathcal{IB} \setminus \{(i^*, \beta^*)\}$, (iv) we go to step 8.

7′ *Select branching rule.* Set $\mathcal{IB} \leftarrow \emptyset$. For each $i \in V_c$ such that $\Delta\hat{a}_i^{\max} > L$, let $\underline{a}_i = \min_{p \in \mathcal{P}_i} \hat{a}_i^p$, $\bar{a}_i = \max_{p \in \mathcal{P}_i} \hat{a}_i^p$ and $\beta_i = (\underline{a}_i + \bar{a}_i)/2$, and set $\mathcal{IB} \leftarrow \mathcal{IB} \cup \{(i, \beta_i)\}$. Sort $\mathcal{IB}$ in decreasing order of $\Delta\hat{a}_i^{\max}$. Select (i^*, β^*) to be the first element of $\mathcal{IB}$, and set $\mathcal{IB} \leftarrow \mathcal{IB} \setminus \{(i^*, \beta^*)\}$.

Finally, it should be mentioned that we generally attempt to follow a best-bound node selection strategy in step 3. However, in order to avoid destroying too many nodes during the backtracking procedure described above, we also recommend the use of a locally breadth-first variant; that is, after a node has been processed, the node we select immediately next is its sibling, if the latter is still in the node queue $\mathcal{N}$, or otherwise proceed with the best-bound selection.

4.4.6 *Algorithmic Details: Initial Upper Bound*

Providing a valid initial upper bound z_{ub} to the branch-and-bound tree in step 1 of the main algorithm of Section 4.4.1 can significantly speed up the solution process by fathoming more nodes of the tree early in the search process, in addition to providing tight upper bounds for the solution of each TSPTW subproblem (see step 1 of the procedure in Section 4.4.3.6). To that end, we run a "template"–based construction heuristic (see, e.g., [140]) at the beginning of the algorithm in an attempt to generate a good initial incumbent ConTSP solution.

The heuristic is based on deriving TSP tours for each period from a template tour, $T_{\text{template}} = \langle 0, v_1, \ldots, v_t, 0 \rangle$, where $t > 1$ and all $v_k \in V_c$ nodes are unique. We consider each of the following tours as template tours: (i) the optimal TSP tour on graph G, i.e., $t = |V_c|$, (ii) the optimal TSP tour on graph G_p, i.e., $t = |V_p|$ and $v_k \in V_p$, $\forall\, k \in \{1, \ldots, t\}$, for each period $p \in \mathcal{P}$, and (iii) the optimal TSP tour involving all customers who require service more than once throughout the planning horizon, namely the set of customers $\{i \in V_c : |\mathcal{P}_i| > 1\}$. For each of these cases, we also consider the template tour obtained by reversing the original template: $\langle 0, v_t, \ldots, v_1, 0 \rangle$. In all, we consider a total of $2|\mathcal{P}| + 4$ template tours as part of our construction heuristic. For each such template tour T_{template}, we attempt to generate a feasible ConTSP solution via the following greedy procedure that appropriately modifies T_{template} so as to obtain tours for each period.

1. *Initialize.* Let $T_p \leftarrow T_{\text{template}}$, for each $p \in \mathcal{P}$, and let cost $z \leftarrow +\infty$.
2. *Generate candidate solution.* For each $p \in \mathcal{P}$:

a) Remove from T_p all customers $i \in T_p$ such that $i \notin V_p$.

b) For each customer $i \in V_p$ such that $i \notin T_p$, insert i in tour T_p at the position that induces the smallest increase in the cost of T_p.

3. *Check feasibility.* If $\{T_1, \ldots, T_{|\mathcal{P}|}\}$ is feasible for ConTSP, set $z \leftarrow \sum_{p \in \mathcal{P}} c(T_p)$.

Assuming the above heuristic has generated at least one feasible ConTSP solution (which is not always guaranteed), we set z_{ub} to be the minimum among the costs of all feasible ConTSP solutions generated, and we adopt the corresponding solution as our initial incumbent.

4.5 COMPUTATIONAL RESULTS

We implemented our algorithms in C++ using the GCC 5.1.0 compiler with optimization level -O2. The runs were conducted on a single-core of an Intel Xeon 2.8 GHz processor with 4 GB RAM. The C Callable Library of CPLEX 12.6 was used to implement all subordinate branch-and-cut algorithms. In the implementation of these branch-and-cut algorithms, all CPLEX-generated cuts and heuristics were disabled because we observed that enabling these features increased computation times; all other options were at their default values. We also selected $\rho = \lceil \bar{n}/30 \rceil$ and $\tau = \rho + \bar{n}/2$ (both in seconds) as the applicable TSPTW solver time limits (see Sections 4.4.3.5 and 4.4.5 for details). Here, $\bar{n}$ is the average number of customers (across time periods) in the ConTSP instance. In all cases, an overall CPU time limit of 2 hours was imposed.

4.5.1 *Test Instances*

Since no benchmark data sets are currently available for the ConTSP in the open literature, we compiled a set of benchmark problems by extending symmetric and asymmetric TSP instances from the well-known TSPLIB library [221] into 3- and 5-period ConTSP instances. In particular, we considered all instances that involve up to 100 customers (i.e., 101 nodes). For each TSPLIB instance, we constructed 3- and 5-period ConTSP instances in a manner similar to that proposed in [140] for the ConVRP. More specifically, we considered each of the n customers to have a probability f of requiring service in each period, where $f \in \{0.5, 0.7, 0.9\}$. Note that, in general, this results in instances in which a different number of customers require service in each time period; however, the average number of customers (across periods) is approximately fn. The distances acquired from the TSPLIB data[11] were interpreted as being both travel costs (c_{ij}) and travel times (t_{ij}), while we assumed all service times (s_i) to be zero. The first node in each of these data sets was arbitrarily regarded to be the depot. For every resulting

11 As per the TSPLIB standard, all distances were rounded down to the nearest integer. Note, however, that this assumption is not restrictive and does not invalidate any of the theoretical results presented in this paper.

multi-period instance, we selected the maximum allowable arrival-time differential, L, to be 10%, 15% or 20% of the maximum (across periods) travel time of the corresponding optimal (single-period) traveling salesman tours.[12] The overall process resulted in nine 3-period and nine 5-period instances for each of the original TSPLIB problems, for a total of 756 ConTSP benchmark instances. These instances are available for download at http://gounaris.cheme.cmu.edu/datasets/contsp/.

Note that the literature on the ConVRP does not allow the vehicle to wait at customer locations and do not consider vehicle duration limits. In order for us to also solve the above instances for the case when waiting is allowed, however, a value for the duration limit D is necessary for each instance. To that end, we set the vehicle duration limit D at 1.1 times the maximum (across periods) travel time of the corresponding optimal (single-period) traveling salesman tours. This value was selected because the tours in the optimal solutions satisfy this duration limit when waiting is not allowed. In this way, all instances are guaranteed to be feasible, allowing us to compare waiting and no-waiting solutions across the whole dataset.

4.5.2 *Tightness of Alternative Formulations and Effect of Valid Inequalities*

In order to gain some insight about the tightness of the three formulations proposed in Section 4.3.1, we compare in Table 4.2 the corresponding root-node gaps, averaged across all instances containing up to 50 customers.[13] The reported gap values constitute an average across all nine 3-period and nine 5-period ConTSP instances constructed from the TSPLIB data indicated in the first column. Furthermore, we report in each case, four characteristic root-node gap quantities, as follows: (C0) the gap obtained using the initial LP relaxation (as described in the preamble of Section 4.3.3), before any cut separation; (C1) the gap obtained after separation of SEC; (C2) the gap obtained after separation of SEC and 2MC; (C3) the gap obtained after separation of SEC, 2MC and IPEC. This analysis helps us appreciate the effect that each of the three families of cuts we consider in this study has on each formulation's tightness. Note that, in order to ensure a fair comparison, we had disabled all CPLEX options pertaining to performing any possible MIP-based bound strengthening.

We observe that the initial LP bounds (gaps C0) are generally weak, in the order of 10% to 20%, but these bounds get significantly reduced, down to roughly 1.6%, when violated SEC are added as cutting planes (gaps C1). The additional separation of 2MC further reduces the gaps by 0.14%, on average. The gaps C2 are indicative of the strength of the lower bounds possible after separating over all the structural, TSP-related cuts.

12 We computed the optimal tours of these TSP instances with an in-house exact TSP solver implementation. We found this to be a rather trivial task, as each single period of a ConTSP instance involved no more than 50 customers.

13 Optimality gaps in this study are defined as $\frac{\text{ub}-\text{lb}}{\text{ub}} \times 100\%$, where ub is the global upper bound and lb is the global lower bound of the branch-and-cut tree. Root-node gaps correspond to optimality gaps after processing the root of the search tree.

Table 4.2: Root-node gaps (%) obtained using each of the proposed formulations after various levels of separation of valid inequalities.

	Formulation 1				Formulation 2				Formulation 3			
Instance	C0	C1	C2	C3	C0	C1	C2	C3	C0	C1	C2	C3
burma14	19.96	0.84	0.84	0.47	12.05	0.84	0.84	0.84	9.21	0.84	0.84	0.84
ulysses16	17.24	0.91	0.91	0.42	11.91	0.91	0.91	0.91	8.45	0.91	0.91	0.86
br17	59.85	0.00	0.00	0.00	32.83	0.00	0.00	0.00	49.20	0.00	0.00	0.00
gr17	22.13	0.73	0.73	0.60	10.98	0.73	0.73	0.73	15.50	0.73	0.73	0.73
gr21	13.24	0.31	0.28	0.21	1.47	0.31	0.28	0.28	7.41	0.31	0.28	0.28
ulysses22	22.41	0.12	0.12	0.04	13.45	0.11	0.11	0.11	12.93	0.12	0.12	0.12
gr24	16.62	0.96	0.91	0.84	5.24	0.96	0.91	0.90	12.37	0.96	0.91	0.90
fri26	20.29	1.58	1.39	1.32	7.71	1.58	1.39	1.39	16.67	1.58	1.39	1.38
bayg29	11.23	1.15	1.12	0.89	5.21	1.15	1.12	1.12	8.06	1.15	1.12	1.12
bays29	14.42	0.56	0.49	0.40	4.64	0.56	0.49	0.49	11.71	0.56	0.49	0.48
ftv33	13.38	4.02	3.81	3.66	8.37	4.00	3.79	3.70	12.10	3.78	3.60	3.60
ftv35	10.94	3.81	3.56	3.50	6.91	3.79	3.55	3.52	9.32	3.74	3.56	3.54
ftv38	8.80	3.84	3.74	3.66	6.10	3.83	3.73	3.66	7.83	3.74	3.63	3.59
dantzig42	24.70	1.35	1.22	1.06	12.11	1.35	1.22	1.22	16.43	1.35	1.22	1.22
swiss42	21.57	1.58	1.33	1.13	6.98	1.58	1.33	1.33	18.80	1.58	1.33	1.32
p43	76.90	0.11	0.11	0.10	52.87	0.11	0.10	0.10	72.70	0.10	0.10	0.10
ftv44	10.69	4.24	3.79	3.79	6.79	4.24	3.80	3.80	9.78	4.23	3.78	3.78
ftv47	8.87	3.19	3.14	3.14	5.03	3.19	3.14	3.14	7.99	3.15	3.11	3.11
att48	22.20	1.42	1.37	1.25	10.79	1.42	1.37	1.37	18.19	1.42	1.37	1.37
gr48	16.58	1.29	0.94	0.88	6.12	1.29	0.95	0.95	13.56	1.29	0.94	0.94
hk48	17.59	1.51	1.39	1.33	5.75	1.51	1.39	1.39	14.37	1.51	1.39	1.39
ry48p	14.86	1.76	1.63	1.63	8.78	1.76	1.63	1.63	12.55	1.67	1.55	1.55
eil51	15.36	2.50	1.66	1.59	4.78	2.50	1.66	1.66	13.55	2.50	1.65	1.65
Avg. (sym.)	18.37	1.12	0.98	0.83	7.95	1.12	0.98	0.98	13.15	1.12	0.98	0.97
Avg. (asym.)	25.54	2.62	2.47	2.43	15.96	2.61	2.47	2.45	22.69	2.55	2.42	2.41
Avg. (all)	20.86	1.64	1.50	1.39	10.73	1.64	1.50	1.49	16.47	1.62	1.48	1.47

Finally, the separation of IPEC further improves the bounds, on average, by 0.11% for Formulation 1 and by 0.01% for Formulations 2 and 3 (gaps C3). Note that the gaps C3 constitute the effective "root-node gaps" of our branch-and-cut framework.

Comparing across formulations, Formulation 1 has the weakest initial LP bound, while the strongest initial LP bound is featured by Formulation 2. However, Formulation 1 appears to be more amenable for bound improvement after separation of valid inequalities. More specifically, after the separation of all three families of cuts considered in this study (gaps C3), the lower bounds of Formulation 1 become stronger and are higher by 0.10% and 0.08%, on average, compared to the respective bounds of Formulations 2 and 3. In 81 out of the 414 instances considered, the final root-node gaps (C3) were equal in all three formulations. In the remaining 333 instances, the root node bound of Formulation 1 was strongest in 215 instances, while those of Formulations 2 and 3 were strongest in 9 and 93 instances, respectively. Table 4.2 also shows the averages across the symmetric and asymmetric instances separately. It is interesting to note that Formulation 3 provided the strongest bound in most of the asymmetric instances, while Formulation 1 gave the strongest bound in most of the symmetric instances. This can be attributed to the fact that, while Formulation 3 is inherently asymmetric, Formulation 1 takes into account any asymmetry only through the IPEC. It remains to be investigated whether the addition of any cuts specific to the asymmetric TSP polytope (see, e.g., [115]) can significantly improve our lower bounds.

Table 4.3 presents the time spent and the number of cuts added at the root node during the runs that resulted in gaps C3; that is, after separation of all three families of cuts (SEC, 2MC and IPEC). Each reported value is averaged across 18 instances (similarly to Table 4.2). The average time spent at the root node is lowest in Formulation 1 and highest in Formulation 3. Thus, Formulation 1 is able to provide a stronger bound in a smaller amount of time, on average. We further observe that, at the root node, the number of IPEC added in the cases of Formulations 2 and 3 is negligible. This is probably because the explicit arrival-time consistency constraints (4.12) and (4.14) in these formulations prevent the frequent occurrence of (fractional) inconsistent paths for us to separate.

4.5.3 *Comparison Between the Algorithms*

In this section, we compare the performance of the branch-and-cut and decomposition algorithms. We only consider the 414 ConTSP instances containing up to 50 customers. Since the branch-and-cut algorithm does not incorporate waiting times or duration limits, we first solve the corresponding instances under the same assumptions, and in order to make a fair comparison, we provide the same initial upper bounds to both algorithms.

In Table 4.4, we compare the decomposition algorithm with the best of the three branch-and-cut algorithms from the previous section, i.e., the one utilizing Formulation 1. For each of the two approaches, we report the number of instances (out of 18) for which optimality was proved as well as the computational time required, averaged across the

Table 4.3: Time spent and number of cuts added at the root node.

	Formulation 1				Formulation 2				Formulation 3			
Instance	t (sec)	SEC	2MC	IPEC	t (sec)	SEC	2MC	IPEC	t (sec)	SEC	2MC	IPEC
burma14	0.01	32.6	0.7	39.9	0.03	14.0	0.0	0.8	0.06	29.9	0.0	0.8
ulysses16	0.01	45.1	2.6	46.3	0.04	21.0	0.3	0.3	0.10	39.3	0.3	0.5
br17	0.00	40.6	0.2	1.3	0.02	30.2	0.1	0.1	0.04	47.4	0.0	0.0
gr17	0.01	51.0	0.0	38.3	0.05	23.6	0.0	0.4	0.15	49.0	0.0	1.3
gr21	0.01	37.7	0.3	48.3	0.06	6.6	0.3	0.3	0.29	37.5	0.3	0.2
ulysses22	0.02	67.2	1.3	42.6	0.09	35.2	0.0	0.1	0.37	63.7	0.0	0.1
gr24	0.03	47.9	3.6	52.4	0.14	15.4	2.2	1.5	0.51	45.9	2.2	2.8
fri26	0.03	61.9	4.4	43.4	0.13	31.4	4.0	0.3	0.67	69.7	4.4	0.6
bayg29	0.05	72.1	4.5	72.4	0.19	25.7	2.3	0.2	0.95	65.6	2.3	0.3
bays29	0.04	68.0	8.3	57.9	0.30	27.9	4.7	0.2	1.21	66.3	4.5	1.1
ftv33	0.05	67.9	27.7	22.3	0.27	50.0	22.8	3.6	1.63	76.8	33.6	0.0
ftv35	0.04	69.4	22.5	14.1	0.20	51.0	22.8	2.1	1.77	76.4	22.8	0.2
ftv38	0.04	62.7	22.7	1.2	0.26	51.4	23.7	0.3	2.94	76.0	39.9	0.6
dantzig42	0.10	113.4	5.2	56.1	0.72	49.2	3.9	0.0	5.61	105.8	3.8	0.0
swiss42	0.17	101.7	17.4	116.6	1.33	43.8	10.8	0.1	6.46	113.8	11.8	0.2
p43	0.51	134.7	46.3	41.7	2.08	89.7	49.4	0.3	80.49	189.9	63.6	0.1
ftv44	0.06	83.7	60.3	0.0	0.43	59.7	63.7	0.0	8.90	87.1	63.0	0.0
att48	0.17	164.1	7.3	73.2	1.45	79.4	4.3	0.0	11.47	161.1	4.6	0.0
ftv47	0.09	70.9	33.4	7.7	0.54	55.1	35.7	3.2	12.71	78.9	35.7	0.0
gr48	0.16	125.9	27.1	54.8	2.52	53.3	24.1	0.1	23.19	137.4	24.3	0.2
hk48	0.11	113.9	17.6	44.7	1.49	49.7	9.4	0.2	13.77	137.6	13.7	0.4
ry48p	0.08	161.9	29.8	0.3	0.60	110.2	28.6	0.1	11.83	174.7	32.9	0.1
eil51	0.40	131.4	72.9	168.5	3.21	53.1	46.9	0.0	29.07	137.9	51.6	0.1
Avg. (sym.)	0.09	82.2	11.5	63.7	0.78	35.3	7.6	0.3	6.26	84.0	8.3	0.6
Avg. (asym.)	0.11	86.5	30.4	11.1	0.55	62.2	30.9	1.2	15.04	100.9	36.4	0.1
Avg. (all)	0.10	83.7	18.1	45.4	0.70	44.6	15.7	0.6	9.31	89.9	18.1	0.4

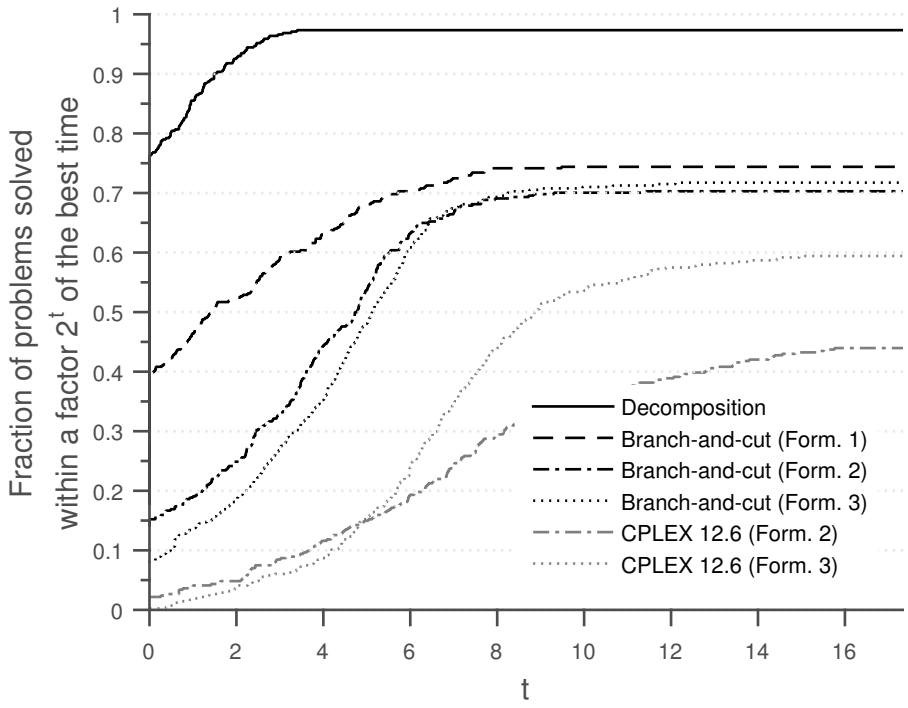


Figure 4.4: Log-scaled performance profiles across 414 ConTSP benchmark instances.

same instances. For those instances which could not be solved within the imposed time limit, the residual gap (defined as $\frac{ub-lb}{ub} \times 100\%$, where ub is the global upper bound and lb is the global lower bound of the branch-and-bound tree) is reported as an average. The two algorithms are also compared in the performance profiles [105] of Figure 4.4. There, we also plot the performance of the branch-and-cut algorithm utilizing the additional two, polynomial-sized MILP formulations, as well as CPLEX 12.6 in its default settings (using general-purpose cuts only, and without separation of the valid inequalities described in Section 4.3.2) to directly address these same formulations. Finally, Table 4.5 presents a summary comparison of the numerical performance of all existing exact methods; the columns have the same meaning as above. The decomposition algorithm is able to prove the optimality of 403/414 instances utilizing an average computation time of 312.9 seconds; of these instances, 95 were unsolved by the best branch-and-cut algorithm. The decomposition algorithm outperforms all branch-and-cut methods, solving more problems and achieving the fastest computation time in almost all instances. Evidently, this algorithm constitutes the new state-of-the-art for solving ConTSP instances to guaranteed optimality.

Table 4.4: Computational comparison of new algorithm against the best branch-and-cut algorithm.

	Decomposition				Branch-and-cut			
	Proven optimal		Residual gap		Proven optimal		Residual gap	
Instance	#	t (sec)	#	Gap (%)	#	t (sec)	#	Gap (%)
burma14	18	0.2	0	–	18	0.1	0	–
ulysses16	18	0.4	0	–	18	0.1	0	–
br17	18	0.0	0	–	18	0.0	0	–
gr17	18	0.2	0	–	18	1.8	0	–
gr21	18	0.1	0	–	18	0.1	0	–
ulysses22	18	0.1	0	–	18	0.1	0	–
gr24	18	2.0	0	–	18	88.7	0	–
fri26	18	6.1	0	–	18	513.4	0	–
bayg29	18	18.4	0	–	17	416.9	1	1.77
bays29	18	1.7	0	–	18	5.1	0	–
ftv33	16	1,039.8	2	0.37	11	3.3	7	1.69
ftv35	18	391.3	0	–	10	1,267.8	8	1.05
ftv38	17	1,496.9	1	1.41	8	323.9	10	1.61
dantzig42	18	23.8	0	–	16	78.9	2	0.41
swiss42	18	96.7	0	–	16	518.8	2	0.49
p43	18	2.7	0	–	0	–	18	0.05
ftv44	13	1,174.2	5	1.27	6	2,496.7	12	1.81
att48	18	624.3	0	–	13	1,091.4	5	0.74
ftv47	17	1,164.1	1	2.05	6	891.8	12	1.25
gr48	18	234.7	0	–	13	353.5	5	0.39
hk48	18	399.0	0	–	11	743.3	7	0.90
ry48p	18	64.7	0	–	13	866.3	5	0.38
eil51	16	960.5	2	1.81	6	1,324.7	12	1.08
All	403	312.9	11	1.29	308	355.3	106	0.98

4.5.4 *Performance of Decomposition Algorithm on Larger Dataset*

We now turn our attention to applying the decomposition algorithm on the expanded benchmark set of 1,512 instances (756 instances with and without waiting). Table 4.6 summarizes its performance. The results are averaged across instances with similar characteristics. In the order in which they appear, the different rows summarize the computational performance on instances with increasing number of customers n, different

Table 4.5: Summary of computational performance of all existing methods.

	Proven Optimal		Residual Gap	
Method	#	t (sec)	#	Gap (%)
Decomposition	403	312.9	11	1.29
Branch-and-cut (Form. 1)	308	355.3	106	0.98
Branch-and-cut (Form. 2)	291	489.9	123	1.03
Branch-and-cut (Form. 3)	297	577.7	117	1.24
CPLEX 12.6 (Form. 2)	182	845.9	232	6.46
CPLEX 12.6 (Form. 3)	246	664.8	168	5.13

number of periods $|\mathcal{P}|$, various levels of f, various levels of L, as well as on instances with and without waiting. This table shows that we are able to solve 1,066/1,512 (71%) instances in an average time of 528.4 seconds. For the remaining instances, the average optimality gap at the time limit was 3.04%. On average, the instances with higher number of periods, mean service frequency $f = 0.7$, lower values of L (higher service quality), and in which waiting is permitted are harder to solve than their counterparts. The decomposition algorithm is able to solve about 96% of all instances with $n \leq 40$ and about 90% of all instances with $n \leq 60$. Interestingly, the average number of branch-and-bound nodes processed exhibits a peak for instances with a number of customers between 40 and 60. This can be explained by the fact that, for $n > 60$, it is mostly the "easier" instances (requiring fewer branch-and-bound nodes) that can be solved within the time limit.

In the following, we present some data related to specific implementation details of the algorithm. Table 4.7 reports the number of TSPTW subproblems solved during the execution of the algorithm, the percentage of subproblems solved with the single-commodity flow formulation (4.4.3.3) (as opposed to those solved with the polyhedral formulation (4.4.3.2); see Section 4.4.3.5), the percentage of subproblems that were feasible, the number of times the node and arc reinsertion heuristics (see Section 4.4.3.6) succeeded in finding a feasible solution (as a percentage of the number of feasible subproblems), as well as the average quality of the obtained heuristic solutions, measured as $\left(z^h - z^*\right)/z^* \times 100\%$, where z^h is the objective value of the heuristic solution and z^* is the objective value of the optimal solution. The table showcases that our TSPTW heuristics succeed in finding a feasible solution in the majority of cases, and that these solutions are of fairly high quality (about 2.5% away from the true optimum, on average). In addition, an increase in instance size leads to more effort being spent in solving the TSPTW subproblems with the branch-and-cut algorithm based on the single-commodity flow formulation (4.4.3.3) rather than with the one based on the polyhedral formulation (4.4.3.2).

Table 4.6: Average computational performance as a function of benchmark characteristics.

		Proven optimal			Residual gap		No incumbent
Parameter	#	#	Nodes	Time	#	Gap (%)	
$n \in (0, 20]$	180	180	50.0	0.3	0	–	0
$n \in (20, 40]$	288	271	386.5	310.4	16	2.36	1
$n \in (40, 60]$	504	422	494.2	637.8	80	2.67	2
$n \in (60, 80]$	216	54	228.2	1,134.3	157	3.33	5
$n \in (80, 100]$	324	139	203.8	1,070.1	185	3.01	0
3-period	756	590	168.9	475.4	166	2.84	0
5-period	756	476	553.1	594.1	272	3.16	8
$f = 0.5$	504	364	337.5	455.9	132	4.73	8
$f = 0.7$	504	310	650.3	621.0	194	2.95	0
$f = 0.9$	504	392	98.2	522.5	112	1.20	0
Low L	504	330	613.5	607.3	168	3.30	6
Med. L	504	358	251.1	520.3	145	2.92	1
High L	504	378	186.7	467.2	125	2.82	1
No Wait	756	542	121.6	563.5	209	3.29	5
Wait	756	524	566.8	492.1	229	2.81	3
All	1,512	1,066	340.5	528.4	438	3.04	8

Table 4.7: Number of TSPTW subproblems solved and effect of heuristic upper bounding.

n	# TSPTWs solved	% Single-commodity flow model	% TSPTWs feasible	% Heuristics succeeded	Heuristic solution quality (%)
$(0, 20]$	35.6	0.0	67.8	78.6	1.02
$(20, 40]$	372.0	21.1	85.5	66.0	3.01
$(40, 60]$	317.2	33.0	87.4	74.0	2.47
$(60, 80]$	400.3	56.6	98.3	82.4	2.52
$(80, 100]$	178.9	70.5	96.9	82.0	2.09
All	276.3	39.5	90.2	75.2	2.54

Table 4.8 reports, as a function of instance size n, the average number of branch-and-bound nodes that had to be processed as well as the average total number of TSPTW subproblems encountered during each search. The table also reports the average time it took to solve TSPTW subproblems that were not encountered in stalled nodes, revealing

that it only took about 1 second, on average, to solve these "well-behaved" subproblems to guaranteed optimality. Finally, the table reports some statistics about the nodes that indeed stalled, namely the percentage of ConTSP benchmarks for which stalling was encountered at least once during the algorithm, and for these instances, the percentage of nodes that actually stalled, the percentage of TSPTW subproblems solved within stalled nodes, and the percentage of total computation time spent inside stalled nodes. It is evident that with increasing problem size n, the likelihood of stalling increases. On average, the algorithm spends 70% of the total CPU time in just 23% of the subproblems. Although this goes beyond the scope of this study, these numbers motivate the need for a more robust TSPTW implementation. For example, one may utilize TSPTW solvers based on dynamic programming, which have reported competitive results [31, 252], although these solvers would be valid only in the case of allowed waiting.

Table 4.8: Number of nodes stalled and time spent inside stalled nodes (averaged across all 1,512 benchmarks).

				Not stalled	Stalled			
n	#	Nodes	TSPTWs	Sol. time TSPTW	%	% Nodes	% TSPTWs	% Time
(0, 20]	180	50.0	48.7	0.01	0.0	0.0	0.0	0.0
(20, 40]	288	1,783.0	1,360.2	0.21	22.6	2.8	5.4	59.6
(40, 60]	504	812.9	637.4	1.02	56.3	9.7	19.0	62.4
(60, 80]	216	923.3	700.8	2.55	93.5	24.8	48.5	73.1
(80, 100]	324	233.2	227.2	7.52	94.4	59.0	90.4	73.6
All	1,512	798.4	626.1	1.26	56.7	12.3	23.5	70.1

As discussed in Section 4.4.6, the initial upper bounds we provide our branch-and-bound search are generated by a simple construction heuristic. In order to assess the potential benefit from employing a better ConTSP heuristic (e.g., a full-fledged metaheuristic) to produce these initial upper bounds, we re-run our algorithm by using our best known solutions as initial incumbents. Table 4.9 summarizes the performance of the algorithm under this setting and compares with the aggregate results from Table 4.6. The experiment reveals that a good initial upper bound can provide tangible numerical benefit. More specifically, with the better initial incumbents, our algorithm was able to solve an additional 124 instances, which constitute approximately 8% of the full benchmark dataset and 27% of the instances that had remained open. In addition, the average optimality gap for the unsolved instances at the time limit was also reduced down to 1.73%.

Table 4.9: Effect of providing the best known solution as an initial incumbent.

Source of initial upper bounds		Proven optimal			Residual gap	
	#	#	Nodes	Time	#	Gap (%)
Construction heuristic	1,512	1,066	340.5	528.4	438	3.04
Best known solutions	1,512	1,190	308.9	413.8	322	1.73

4.5.5 *The Price of Consistency*

In this section, we aim to estimate the additional cost that one must incur, on average, in order to provide consistent service. To that purpose, we report in Table 4.10 the difference between the optimal ConTSP cost and the sum of the minimum costs of the corresponding traveling salesman tours in each time period (without consistency), as a percentage of the latter. This quantity is reported separately for each different level of service frequency (f), length of the planning horizon (h) and maximum allowable arrival-time differential (L) considered in this study, in order to also investigate the effect of these parameters on the overall cost increase.

Table 4.10: Cost of providing consistent service.

	Low L	Med. L	High L
$f = 50\%$	1.85	1.32	1.09
$f = 70\%$	2.10	1.64	1.37
$f = 90\%$	0.99	0.75	0.67
$h = 3$	1.52	1.18	1.03
$h = 5$	1.77	1.30	1.06
	1.65	1.24	1.05

Our study shows that an arguably small cost increase, 1.31%, must be incurred, on average, in order to provide consistent service. Across all instances we considered, the required additional cost varies between 0% to 8.25%, with four out of five instances commanding less than 2% of a cost increase. For low values of the maximum allowable arrival-time differential, i.e., high levels of service consistency, the additional cost that must be incurred reaches 1.65%, on average, and this number decreases to 1.05% when consistency levels are reduced (high L). Moreover, it is interesting to observe that a relatively higher cost must be incurred when the service frequency is 70%, as opposed to when it is 50% or 90%, and when the number of time periods is 5, as opposed to when it is 3. We also remark that the price of consistency is, on average, higher for asymmetric instances.

We remark that the selection of the above levels of L, the maximum allowable arrival time differential, is not arbitrary and in fact, these levels correspond to consistency requirements which are tight, in some sense. This is because, on average, the maximum arrival time difference (across all customers) of the multi-period solution consisting of the optimal traveling salesman tours in each period, is roughly 68% of the maximum vehicle travel time (across periods).[14] Therefore, for the benchmark problems we considered, arrival time consistency is not enforced using a purely cost-based routing approach and levels of $L = 10\%$, 15% and 20% are reasonably tight. Moreover, assuming these optimal traveling salesman tours correspond to an "8-hour shift of driving", a value of $L = 10\%$ roughly corresponds to a maximum arrival time difference of 48 minutes (i.e., a guaranteed arrival time within 24 minutes of a nominal time), while $L = 20\%$ corresponds to a maximum arrival time difference of 1 hour 36 minutes (i.e., a guaranteed arrival time within 48 minutes of a nominal time). We believe these values are reasonable from a practical point of view.

4.5.6 *Cost Savings due to Allowing Waiting*

The price of consistency computed in the previous section can be decreased by allowing the vehicle to wait before providing service. In this section, we aim to estimate the cost savings that are possible by allowing the vehicle to do so. More specifically, we define the *cost savings*, $S = 100\% \times (z_{NW} - z_W) / z_{NW}$, where z_W is the best known cost of the ConTSP with waiting and z_{NW} is the best known cost of the ConTSP without waiting. We also define the *consistency premium recovery ratio*, $R = 100\% \times (z_{NW} - z_W) / (z_{NW} - z_{TSP})$, where z_{TSP} is the sum of the minimum costs of the corresponding traveling salesman tours in each time period. In this context, z_{TSP} reflects the cost that the operator would anyways have to incur even if there was no requirement for service consistency. The cost savings and recovery ratio are reported separately for each different level of service frequency (f), length of the planning horizon ($|\mathcal{P}|$) and maximum allowable arrival-time differential (L) considered in this study, in order to also elucidate the effect of these parameters on the overall cost savings.

For 51 instances in our dataset, the objective value of the optimal ConTSP solution (with or without waiting) is equal to the objective value of the optimal, purely cost-based routing plan without consistency. For 341 out of the remaining 705 instances, the objective value of the ConTSP is the same with or without waiting, i.e., no cost savings are possible. For the remaining 364 instances for which the cost recovery is non-zero, about 0.6% of the cost incurred when not allowing waiting and about 40.1% of the increase with respect to the purely cost-based routing plan can be recovered, on average, by allowing the vehicle to wait. Across all instances we considered, the savings varies between 0% and 6.5%,

14 For ConTSP instances with symmetric costs, this number was obtained by minimizing the maximum arrival time difference across all 2^h configurations of solutions with the same cost.

Table 4.11: Average cost recovery across all 756 ConTSP benchmark instances.

Parameter	#	# $z_{NW} > z_{TSP}$	# $z_{NW} > z_W$	S (%)	R (%)
$n \in (0, 20]$	90	63	26	0.5	52.8
$n \in (20, 40]$	144	132	84	0.9	46.6
$n \in (40, 60]$	252	243	150	0.4	33.4
$n \in (60, 80]$	108	107	40	0.5	41.4
$n \in (80, 100]$	162	160	64	0.5	41.4
3-period	378	341	170	0.5	40.5
5-period	378	364	194	0.6	39.7
$f = 0.5$	252	236	155	0.6	42.5
$f = 0.7$	252	235	130	0.6	35.7
$f = 0.9$	252	234	79	0.3	42.5
Low L	252	239	136	0.8	38.2
Med. L	252	235	114	0.4	38.6
High L	252	231	114	0.4	43.9
All	756	705	364	0.6	40.1

while the consistency premium recovery ratio varies between 0% and 100%, with one out of every four instances enjoying an excess of 0.3% savings and 30% recovery.

4.6 SUMMARY

Multi-period routing problems with consistency requirements represent a practically-relevant class of problems, as distributors can gain significant competitive advantages by providing consistent service to their customers. Arrival-time consistency, i.e., the requirement to visit customers at approximately the same time during the routing horizon, has been identified as one plausible avenue to add such value. In this chapter, we introduced two exact algorithms for the Consistent Traveling Salesman Problem. These constitute the first exact approaches in the open literature that address a routing problem with consistency constraints.

The first algorithm was based on applying branch-and-cut on three mixed-integer linear programming formulations. The formulation that uses only binary variables and that relies on cutting planes to enforce all consistency requirements was shown to be the most attractive from a computational viewpoint. The second algorithm was based on a decomposition idea. It is superior to the best branch-and-cut algorithm, being able to solve to guaranteed optimality benchmark instances with up to 100 customers requiring service over a 5-period horizon. Moreover, it allowed for the possibility of the vehicle

to idle (wait) at customer locations in addition to considering–whenever applicable–route duration limits, overcoming restrictive assumptions that were made by previous approaches.

Our study suggests that a modest routing cost increase of the order of 1-2% would typically suffice so as to provide consistent service, and that up to 40% of the cost increase that would be incurred in order to provide consistent service can be recovered by allowing the vehicle to idle en route. Expected benefits for the distributor, however, may well make up for this small cost increase. Evidently, consistency of service constitutes a value proposition that distributors should consider further.

4.7 APPENDIX: NOMENCLATURE

n	Number of customers
h	Number of time periods
$\mathcal{P}$	Set of time periods
$G = (V, A)$	Complete directed graph with node set V and arc set A
c_{ij}	Routing cost along arc $(i, j) \in A$
t_{ij}	Travel time along arc $(i, j) \in A$
V_c	Set of customers
$\mathcal{P}_i$	Subset of time periods in which customer $i \in V_c$ requires service
V_p	Subset of customers requiring service in period $p \in \mathcal{P}$
A_p	Subset of arcs covering customers in V_p
T_p	Traveling salesman tour on graph $G_p = (V_p \cup \{0\}, A_p)$
$c(T_p)$	Cost of the traveling salesman tour T_p
a_i^p	Arrival time at customer $i \in V_p$ in period $p \in \mathcal{P}$
D	Route duration limit
L	Maximum allowable arrival-time differential
$\mathcal{TW}_i = [e_i, \ell_i]$	Time window associated with customer $i \in V_c$
$N_p^+(i)$	Set of nodes j for which there is an arc from i to j in arc set A_p
$N_p^-(i)$	Set of nodes j for which there is an arc from j to i in arc set A_p
$A(S)$	Subset of arcs with both end points in $S \subseteq V_c$
$\delta(S)$	Subset of arcs with exactly one end point in $S \subseteq V_p$ and the other in $V_p \cup \{0\} \setminus S$
x_{ijp}	Binary variable $\in \{0, 1\}$ indicating if arc $(i, j) \in A$ is used in the tour of period $p \in P$
α_{ip}	Continuous variable recording the arrival time at customer $i \in V_p$ in period $p \in P$
y_{ijp}	Continuous variable recording the arrival time at customer $i \in V_p$ in period $p \in P$ if $x_{ijp} = 1$, and 0 otherwise
P_{CONTSP}	Convex hull of integer feasible solutions of Formulation 1

5

STRATEGIC ALLOCATION OF TIME WINDOWS

The previous chapters illustrated the benefits of considering uncertainty in vehicle routing at the operational and tactical levels. At these time scales, several parameters remain unchanged over the duration of the planning horizon, as they are exogenously determined by longer-term decisions, and they can therefore be treated deterministically. Typical examples include the allocation of customers to depots as well as of service time windows for each customer on a particular day. This chapter addresses the strategic decision-making problem of assigning values to such parameters, and treats them as endogenous to the problem.

We focus our study to the long-term allocation of a weekly service day and daily time window for each customer. nce such a time window has been allocated, the distributor must contractually adhere to it on a daily basis. Since the time windows control the structure of feasible vehicle routes, their strategic allocations end up strongly influencing the operationally incurred transportation costs. Determining the time windows is therefore an important albeit non-trivial task, since operational level information (such as travel times or customer demand) is often not known with certainty at the strategic level when they are to be allocated.

In this chapter, we show that the strategic problem of allocating service time windows is more or less equivalent to the tactical problem of enforcing service consistency, which was the subject of the previous chapter. We leverage this equivalence to develop a highly flexible yet efficient scenario-based approach to tackle the time window allocation problem. From a modeling viewpoint, the approach can accommodate scenario-based models of uncertainty for any routing-specific parameter. From an algorithmic viewpoint, it can utilize any available (exact or heuristic) vehicle routing solver as a black box. And from a practical viewpoint, it allows decision-makers to quantify the trade-off between modeling effort and expected cost savings when considering a larger number of future scenarios during strategic time window assignment.

This chapter is organized as follows. After the problem is further motivated in Section 5.1, Section 5.2 reviews existing papers that study related problems. Section 5.3 then provides a general mathematical model of the problem while Sections 5.4 and 5.5 describe our solution approach. In Section 5.6, we conduct an extensive computational study on

existing as well as new datasets, and in Section 5.7, we close with a summary of the results.

5.1 MOTIVATION

The commitment to deliver (or pickup) goods within scheduled time windows is a common practice in several real world distribution networks. In many industries, these time windows are mutually agreed upon by the distributor and customer through long-term delivery contracts. For example, in a distribution network of retailers, it is common that deliveries to a retail store are always made on the same day of the week (at about the same time) for an entire year [242, 265]. Likewise, in maritime distribution of liquefied natural gas, a central planning activity is to design and negotiate contractual agreements of *annual delivery plans* that specify delivery dates and corresponding delivery quantities to customers [277]. From the customer's point of view, this is crucial for efficient inventory management and scheduling of personnel to process the delivery. From the distributor's point of view, it reduces the variability across repetitive deliveries and exposes efficiencies that add up to significant cost savings. Short- and medium-term contracts of similar nature can be also found in small-package shipping where, for instance, courier companies provide a delivery time window to customers receiving sensitive packages [157]. Other examples of applications where such operations are typical include, among others, attended home delivery in e-commerce businesses [4] and internet installation services [259].

Once a time window has been agreed upon and communicated to the customer, the distributor must attempt to meet it on an operational (e.g., daily) basis as well as possible. This is done by solving a Vehicle Routing Problem with Time Windows (VRPTW) to determine a delivery schedule that adheres to the agreed time windows. The assigned time windows strongly influence the structure of feasible delivery schedules and, hence, the daily incurred distribution costs. Therefore, a natural choice is to assign time windows based on the arrival times at customer locations in the optimal (i.e., minimum cost) vehicle routing schedule. However, this seemingly optimal decision may become highly suboptimal in the presence of operational uncertainty.

In reality, operational level information (such as customer demands or travel times) is often not known with certainty at the strategic level when time windows are to be decided. For example, the demand volume of a customer typically fluctuates per delivery. Similarly, travel times vary on a day-to-day basis (e.g., because of unpredictable traffic conditions). The true values of these operational parameters are not known far in advance, and often may become known only on the day of delivery before the vehicles are dispatched. This makes the strategic assignment of time windows a non-trivial task. Indeed, if one utilizes only nominal values of the uncertainty when assigning time windows, then it will often lead to situations in which the distribution costs are unacceptably high, since the nominal delivery schedule may no longer be feasible, let alone optimal, in such

cases. Fortunately, with the increasing availability of data, distributors can readily obtain forecasts of uncertain operational parameters (e.g., as perturbations from their nominal values). It is possible to take advantage of this information and assign time windows in a way that will lead to low distribution costs in the long run. The goal of this chapter is to study the problem of strategic time window assignment in the presence of operational uncertainty.

This chapter builds upon the work of [242], which introduced the Time Window Assignment Vehicle Routing Problem (TWAVRP). The TWAVRP consists of assigning time windows of pre-specified width within some exogenous time windows to a set of known customers. The exogenous time windows typically correspond to operating hours of the customer but may also arise from hours-of-work or other government regulations. The work of [242] studies the TWAVRP under situations in which the demand volume of the customers is unknown and subject to uncertainty. However, a finite set of "scenarios," each describing a possible realization of demand for every customer, is assumed to be given with known probability of occurrence. This information is used to formulate a two-stage stochastic program, in which the first-stage decisions are to assign time windows, while the second-stage decisions are to design vehicle routing schedules satisfying the assigned time windows, one for each of the demand scenarios. The objective is to minimize the total routing costs, averaged over the postulated scenarios. A similar modeling approach is followed in [241], with the only difference that the first-stage time windows are selected from a finite set of *a priori* constructed windows; this problem is referred to as the *discrete* TWAVRP to distinguish it from the original *continuous* TWAVRP. In this work, we consider both cases, and we shall in fact allow also for the generalized case in which feasible time window assignments lie in a continuous set for some portion of the customer base and in a discrete set for the remaining portion.

Algorithms to solve the aforementioned stochastic programming models have been proposed in [94, 242] for the continuous version, and in [241] for the discrete version of the problem. The algorithms of [241, 242] are based on branch-price-and-cut and can solve instances with 25 customers and 3 demand scenarios to optimality, while the algorithm of [94] is based on branch-and-cut and can address instances containing 40 customers and 3 scenarios. Several heuristics have also been proposed in [241] for the discrete setting that can address instances containing up to 60 customers. Recently, [240] studied a variant of the TWAVRP with time-dependent travel times and proposed a branch-price-and-cut algorithm that can solve instances with 25 customers and 3 demand scenarios.

A problem that is closely related to the strategic TWAVRP is the Consistent Vehicle Routing Problem (ConVRP) [140], which is motivated in the context of operational level planning. The ConVRP aims to design minimum cost vehicle routes over a finite, multi-day horizon to serve a set of customers with known demands. The goal is to design routes that are *consistent* over time; this translates to satisfying any of the following requirements each time service is provided to a customer: *(i)* arrival-time consistency, wherein the customer should be visited at roughly the same time during the day, *(ii)*

person-oriented consistency, in which the customer should be visited by the same driver, and whenever applicable, *(iii)* load consistency, for which a customer should receive roughly the same quantity of goods. We refer the reader to [172] for an overview of this problem and its applications.

Conceptually, the assigned time windows in the TWAVRP (which are also referred to as the *endogenous time windows*) serve to satisfy the arrival-time consistency requirement of the ConVRP, which requires that every customer be visited at roughly the same time whenever service is requested. Formally, the ConVRP requires that the difference between the earliest and the latest arrival times at each customer location must differ by no more than some pre-specified constant bound, which is referred to as the *maximum allowable arrival-time differential*. This bound is analogous to the pre-specified width of the endogenous time window in the continuous TWAVRP, and this equivalence between the two problems has been previously acknowledged in [240, 242].

The equivalence between the TWAVRP and the arrival-time ConVRP has two important consequences. First, we observe that, in the most general case, the ConVRP allows for the possibility that not all customers require service in all time periods and that operational parameters (such as customer demands or travel times) differ from one time period to the other. Translated in the context of the TWAVRP, this allows us to address applications in which a fraction of the customer base does not require frequent (e.g., daily) service (by considering scenarios where certain subsets of customers have no demand), as well as to treat uncertainty in a wider variety of parameters such as travel and service times (by considering scenarios in which their values represent perturbations from some nominal value). We therefore study the TWAVRP under a more general definition than what has been previously considered in the literature, in which it is possible to incorporate uncertainty in several operational parameters at once. However, we do remark that, as is the case with traditional stochastic programming models, the simultaneous treatment of uncertainty in several parameters may come at the cost of an explosion in the number of scenarios that have to be considered.

The second consequence of the equivalence between the TWAVRP and the arrival-time ConVRP is that any algorithm developed for the latter can be used to obtain solutions for the former. In this work, we adapt the decomposition algorithm proposed in Chapter 4 for the Consistent Traveling Salesman Problem (ConTSP), the single-vehicle variant of the ConVRP that focuses purely on the aspect of arrival-time consistency, to obtain a new algorithm for the TWAVRP. Our method can be viewed as a *scenario decomposition algorithm* in the language of stochastic programming, and is not based on branch(-price)-and-cut that has been the de facto approach for solving TWAVRP models. Our algorithm has the attractive features of *modularity* and *scalability*. It is modular in accommodating *(i)* continuous and discrete time windows, *(ii)* any VRPTW solver (exact or heuristic), *(iii)* routing-specific constraints (e.g., heterogeneous fleets), and *(iv)* generic scenario descriptions. Moreover, it can be readily parallelized which allows postulating a large number of scenarios of the uncertainty. Together with *(ii)* above, this means that our

algorithm is also scalable. The distinct contributions of our work may be summarized as follows.

5.2 RELATED LITERATURE

This section reviews papers in the vehicle routing literature, other than those mentioned in the introduction, which deal with aspects of *(i)* endogenously imposed time windows, *(ii)* consistent service considerations, and *(iii)* stochastic or uncertain parameters. We choose not to review the extensive literature on the VRPTW; instead, we refer interested readers to [102].

The authors of [157] study a time window assignment problem that is encountered by courier companies who must quote delivery time windows to customers receiving sensitive packages. In this problem, which they refer to as the VRP with Self-Imposed Time Windows, travel times are uncertain but all customers and their demands are known *a priori*. Using this information, the service provider must simultaneously determine *(i)* a single routing plan to serve all customers, and *(ii)* time window assignments that will be quoted to the customers before the vehicles depart from the depot. The objective is to minimize the sum of (deterministic) routing costs and (expected) overtime and tardiness penalty costs. Since travel times are uncertain, the key challenge is to determine the optimal *placement* of time windows (along each vehicle route) in step *(ii)* so as to avoid penalties due to delays. The uncertainty in travel time along each arc is modeled via a discrete set of "disruption" scenarios (each representing a deviation from some nominal value). Under the assumption that at most one arc will be disrupted on any vehicle route, the authors propose a Tabu Search heuristic for route generation and a linear programming approach for time window placement that inserts "time buffers" along each vehicle route. Recently, with a goal to solve the same problem, the authors of [262] extended the work of [157]. On the one hand, they relax the assumption that only one arc will be disrupted and use probabilistic chance constraints to guarantee reliable service. On the other hand, they propose an alternative model of uncertainty in which the stochastic deviations in travel times are modeled as continuous gamma-distributed random variables. Finally, by considering also the width of the time window (along with its placement) as a decision variable, they propose an Adaptive Large Neighborhood Search solution procedure.

A related line of work is the so-called Time Slot Management Problem that is motivated in the context of attended home delivery in e-commerce businesses [4]. Here, customers place online orders for products (e.g., groceries) and, during this process, they select one time window (amongst a number of available ones) in which they want their product to be delivered. From the service provider's point of view, the challenge is to design a finite set of time windows (instead of just one) to offer to potential customers in different zip code areas. The problem is complicated by the fact that, during the design phase, the set of customers as well as their demand is not known with certainty. The objective is to

design time windows that would not only yield low distribution costs in the long run, but also satisfy marketing or regulatory requirements. In general, existing approaches (e.g., see [4, 67, 150]) deal with uncertainty by simply using expected values of customer demand whose temporal distribution is either assumed to be uniform over the offered time windows or determined via simulations. The expected routing cost associated with a candidate set of time windows is then estimated via coarse continuous approximation methods (e.g., see [93, 111]) or detailed vehicle routing models. These estimates are embedded within some heuristic procedure (e.g., local search) to determine the final set of time windows. We refer the reader to [3] for an overview of research problems in the area of attended home delivery, including time slot management. Finally, we mention the work of [259] who also study a time window assignment problem that is motivated in the context of home-attended services (e.g., cable installation). Here, as is the case in attended home delivery, customers dynamically place orders for some service. However, instead of the customer choosing a time window from a number of available ones, the service provider must quote a service time window to the customer at the time of request. Similar to the VRP with Self-Imposed Time Windows, travel and service times are stochastic and the objective is to minimize expected delays. However, unlike the latter problem, not all customers who will be serviced are known at the time when a particular request is received, and thus the customer base is also stochastic. The authors use approximate dynamic programming techniques to obtain time window assignments in real time.

We conclude our literature review by mentioning the relationship of the TWAVRP to Stochastic Vehicle Routing Problems (SVRP). The latter class of problems also treats parameter uncertainty in the context of vehicle routing. However, unlike the TWAVRP which is inherently a strategic decision-making problem, the SVRP is an operational problem. Specifically, in the TWAVRP, the exact values of all parameters are assumed to be known before the vehicle routes are to be determined on a particular day. In contrast, in the SVRP, the vehicle routes must be determined before the parameter values become known, which are only gradually revealed during the execution of the routing plan. This requires fundamentally different modeling considerations and corresponding solution approaches. The most common modeling paradigms are *(i)* recourse models, *(ii)* reoptimization models, and *(iii)* chance-constrained models. In *(i)*, a planned solution is designed in the first stage and recourse actions based on a predetermined policy are taken in the second stage when the uncertainties are revealed. For example, the capacity of a vehicle may get exceeded *en route*, if demands are stochastic at the time of vehicle dispatch; in such cases, a recourse policy, such as a detour to the depot to empty the vehicle, must be explicitly incorporated in the model [107, 123]. In *(ii)*, the planned solution is dynamically modified as the uncertain parameters (e.g., demands or travel times) become gradually revealed during the execution of the vehicle routes [226]. Finally, in *(iii)*, probabilistic or chance constraints are used to explicitly control the level of risk that is acceptable to the decision-maker [179]. We refer interested readers to [125],

who provide an excellent overview of applications, models and solution algorithms for the SVRP and its variants.

5.3 PROBLEM DEFINITION

Let $G = (V, A)$ denote a directed graph with nodes $V = \{0, 1, \ldots, n\}$ and arcs A. Node $0 \in V$ represents the unique depot, and each node $i \in V_C := V \setminus \{0\}$ represents a customer. The operating hours of the depot are represented by the time window $[e_0, \ell_0]$, where an unlimited number of vehicles, each of capacity Q, are available for service. Each vehicle incurs a transportation cost $c_{ij} \in \mathbb{R}_+$ and a travel time $t_{ij} \in \mathbb{R}_+$ if it traverses the arc $(i, j) \in A$. Furthermore, each customer $i \in V_C$ features a demand $q_i \in \mathbb{R}_+$, service time $u_i \in \mathbb{R}_+$ and *exogenous* time window $[e_i, \ell_i]$ (e.g., representing operating hours). The key decisions in the TWAVRP are to decide the *endogenous* time windows $\tau_i \in TW_i$ to be assigned to each customer $i \in V_C$. The definition of the feasible time window set TW_i may be either of the following (refer to Figure 5.1):

- In the *continuous* setting, the assigned time window must have a pre-specified width $w_i \in \mathbb{R}_+$; that is, $TW_i = \{[y_i, y_i + w_i] : e_i \leq y_i \leq \ell_i - w_i\}$, where we assume, without loss of generality, that $e_i \leq \ell_i - w_i$.
- In the *discrete* setting, the assigned time windows must belong to a pre-specified finite set; that is, $TW_i = \left\{[\underline{y}_{i1}, \overline{y}_{i1}], \ldots, [\underline{y}_{iN_i}, \overline{y}_{iN_i}]\right\}$, where we can assume, without loss of generality, that all N_i candidate time windows are pairwise either non-overlapping or partially overlapping.[1] Therefore, the set TW_i can be ordered so that $e_i = \underline{y}_{i1} \leq \ldots \leq \underline{y}_{iN_i}$ and $\overline{y}_{i1} \leq \ldots \leq \overline{y}_{iN_i} = \ell_i$.[2]

We denote by $V_{\text{cont}} \subseteq V_C$ and $V_{\text{disc}} \subseteq V_C$ the subset of customers whose feasible time window sets are continuous and discrete respectively. We note that $V_{\text{cont}} \cap V_{\text{disc}} = \emptyset$ and $V_{\text{cont}} \cup V_{\text{disc}} = V_C$.

In practice, operational parameters such as those related to the transportation network (costs c, travel times t) or the customers (demands q, service times u) are often not known with certainty at the strategic level when time windows must be allocated. Let θ denote the set of all operational parameters, and let $\mathbb{P}$ denote the joint probability distribution of θ. The goal of the TWAVRP is to assign the time windows $\tau_i \in TW_i$ in a way that minimizes the expected cost of routing:

$$\begin{aligned} &\underset{\tau}{\text{minimize}} \;\; \mathbb{E}_{\theta \sim \mathbb{P}} \left[\texttt{VRPTW}(\tau; \theta)\right] \\ &\text{subject to} \;\; \tau_i \in TW_i \quad \forall i \in V_C. \end{aligned} \tag{5.1}$$

1 Two completely overlapping time windows $[a, b]$ and $[c, d]$ with $a \leq c \leq d \leq b$ can be replaced with the larger of the two time windows $[a, b]$.

2 We remark that $e_i = \underline{y}_{i1}$ and $\ell_i = \overline{y}_{iN_i}$ can be achieved by preprocessing. If $e_i < \underline{y}_{i1}$, then e_i can be shifted forward to match $\underline{y}_{i1}$, and if $e_i > \underline{y}_{ib}$, then $\underline{y}_{ib}$ can be shifted forward to match e_i. A similar argument applies for ℓ_i.

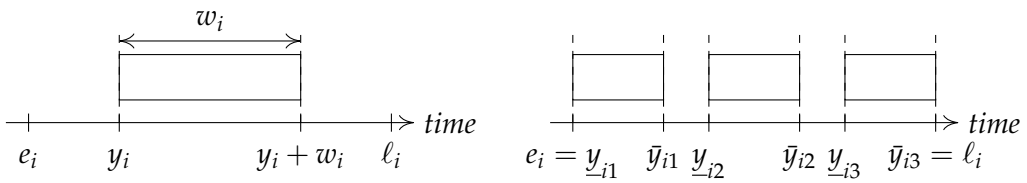


Figure 5.1: Illustration of continuous (left) and discrete (right) time window sets TW_i. In the continuous case, the assigned time window can be any sub-interval of $[e_i, \ell_i]$ of length w_i, while in the discrete case, the assigned window must be one of the intervals $[\underline{y}_{i1}, \bar{y}_{i1}]$, $[\underline{y}_{i2}, \bar{y}_{i2}]$ or $[\underline{y}_{i3}, \bar{y}_{i3}]$.

In the above *stochastic programming* formulation, $\texttt{VRPTW}(\tau;\theta)$ denotes the minimum cost of the vehicle routing problem with time windows τ_i, $i \in V_C$ and operational parameters θ. A formal mathematical definition of $\texttt{VRPTW}(\tau;\theta)$ follows.

5.3.1 *Mathematical Definition of* $\texttt{VRPTW}(\tau;\theta)$

In this section, the time windows τ and operational parameters θ are assumed to be fixed to certain values in their domain and support respectively. As far as the routing operation is concerned, we shall assume that each customer with non-zero demand must be visited exactly once by a single vehicle; that is, split deliveries are not allowed. In this regard, a *route set* $\mathbf{R} = (R_1, \ldots, R_m)$, where $m \geq 1$, represents a partition of the customer set V_C. Here, $R_k = (R_{k,1}, \ldots, R_{k,n_k})$ represents the k^{th} vehicle route, $R_{k,l}$ the l^{th} customer and n_k the number of customers visited by vehicle k. The cost of a route R_k is evaluated as $c(R_k) = \sum_{l=0}^{n_k} c_{R_{k,l},R_{k,l+1}}$, where we define $R_{k,0} = R_{k,n_k+1} = 0$, and the cost of $\mathbf{R}$ is defined as $c(\mathbf{R}) = \sum_{k=1}^{m} c(R_k)$. The route set $\mathbf{R}$ is feasible, if (i) all capacity constraints are satisfied, i.e., $\sum_{i \in R_k} q_i \leq Q$ for all $k \in \{1, \ldots, m\}$, and (ii) all time window constraints are satisfied, i.e., there exists a vector of arrival times, $\boldsymbol{a} \in \mathcal{X}(\mathbf{R}, \tau; \theta)$, where $\mathcal{X}(\mathbf{R}, \tau; \theta)$ is the feasible solution set of the following linear system of inequalities:

$$\mathcal{X}(\mathbf{R}, \tau; \theta) = \left\{ \boldsymbol{a} \in \mathbb{R}_+^n \;\middle|\; \begin{array}{ll} a_{R_{k,1}} \geq e_0 + t_{0,R_{k,1}} & \forall k \in K := \{1, \ldots, m\} \\ a_{R_{k,l+1}} - a_{R_{k,l}} \geq t_{R_{k,l},R_{k,l+1}} + u_{R_{k,l}} & \forall l \in \{1, \ldots, n_k - 1\}, \forall k \in K \\ a_{R_{k,n_k}} \leq \ell_0 - t_{R_{k,n_k},0} - u_{R_{k,n_k}} & \forall k \in K \\ a_i \in \tau_i & \forall i \in V_C \end{array} \right\} \tag{5.2}$$

In this definition, $a_{R_{k,l}}$ denotes the arrival time at location $R_{k,l}$, i.e., the arrival time at the l^{th} location on the k^{th} vehicle route. The first three inequalities essentially require that the arrival time at any location must be at least as large as the sum of the arrival and service times in the previous location and the time to travel from the previous to the current location. The last inequality requires the arrival time at customer location $i \in V_C$ to be within the time window τ_i. Observe that, by this definition, if a vehicle arrives

at customer location $i \in V_C$ at a time earlier than $\underline{\tau}_i := \min_{t \in \tau_i} t$, then it is allowed to wait until $\underline{\tau}_i$. However, arriving later than $\overline{\tau}_i := \max_{t \in \tau_i} t$ is not permitted. We denote by $\mathcal{R}(\tau; \theta)$ the set of all feasible route sets for the given realization of operational parameters θ and time window assignment τ. The value of $\texttt{VRPTW}(\tau; \theta)$ can now be defined as the optimal value of the following optimization problem:

$$\begin{aligned} &\underset{\mathbf{R}}{\text{minimize}} && c(\mathbf{R}) \\ &\text{subject to} && \mathbf{R} \in \mathcal{R}(\tau; \theta). \end{aligned} \tag{$\texttt{VRPTW}(\tau; \theta)$}$$

5.3.2 *Deterministic Equivalent of Stochastic Programming Formulation*

In practice, the exact joint probability distribution $\mathbb{P}$ is either not explicitly available or is hard to obtain. Indeed, even if it is known exactly, computing the objective function involves multi-dimensional integration of the function $\texttt{VRPTW}(\tau; \cdot)$, which is practically impossible considering that the solution of the deterministic problem $\texttt{VRPTW}(\tau; \theta)$ is itself challenging and cannot be obtained in closed form. Instead, we assume that we are given a finite set of S scenarios $\theta_1, \ldots, \theta_S$ along with associated probabilities of occurrence $p_1, \ldots, p_S$, where $p_s > 0$, $s \in \mathcal{S} := \{1, \ldots, S\}$ and $\sum_{s \in \mathcal{S}} p_s = 1$. In this situation, we seek to optimize the following *deterministic equivalent* of problem (5.1), obtained by replacing the expectation with a sample average:

$$\begin{aligned} &\underset{\tau}{\text{minimize}} && \sum_{s \in \mathcal{S}} p_s \texttt{VRPTW}(\tau; \theta_s) \\ &\text{subject to} && \tau_i \in TW_i \quad \forall i \in V_C. \end{aligned} \tag{5.3}$$

Following the definition of $\texttt{VRPTW}(\tau; \theta)$, the above sample average formulation (5.3) can be equivalently represented as follows:

$$\begin{aligned} &\underset{\tau, \mathbf{R}}{\text{minimize}} && \sum_{s \in \mathcal{S}} p_s c(\mathbf{R}_s) \\ &\text{subject to} && \tau_i \in TW_i \quad \forall i \in V_C \\ & && \mathbf{R}_s \in \mathcal{R}(\tau; \theta_s) \quad \forall s \in \mathcal{S}. \end{aligned} \tag{5.4}$$

The optimization problem (5.4) shall be our primary focus for the rest of the chapter. We shall denote by $(\tau, \{\mathbf{R}_s\}_{s \in \mathcal{S}})$ a feasible solution to this problem.

5.4 SOLUTION APPROACH

Our solution approach for the TWAVRP is motivated by the observation that, in the continuous setting (where $V_{\text{disc}} = \emptyset$), problem (5.4) can be reduced to an instance of the arrival-time ConVRP. Consequently, any algorithm to solve the latter class of problems can be used to solve problem (5.4). Section 5.4.1 presents an exact branch-and-bound

algorithm for this purpose; we note, however, that the presented algorithm is more general-purpose, since it can also address the setting where $V_{\text{disc}} \neq \emptyset$. Section 5.4.2 presents new valid disjunctions that can be used as alternative branching rules in the algorithm; Section 5.4.3 presents upper bounding procedures (i.e., generating good time window assignments) in the context of our algorithm; and finally, Section 5.4.4 shows how the (exact) algorithm can be modified as a heuristic to solve large-scale instances.

5.4.1 *Overview of Exact Algorithm*

We adapt the decomposition algorithm of Chapter 4 developed for the Consistent Traveling Salesman Problem (the single-vehicle variant of the ConVRP) to solve the TWAVRP to optimality. Notably, we extend the algorithm to incorporate also the case of discrete time windows. This section presents the main ingredients of the algorithm translated into the TWAVRP context.

The algorithm uses a branch-and-bound tree search to identify the optimal time window assignments by solving within each node a set of VRPTW instances. The tree is initialized with the original problem instance enforcing only the exogenous time windows $[e, \ell]$. If valid time window assignments $\tau \in TW$ cannot be constructed using the optimal solution of the current node, the algorithm creates new nodes by using disjunctions (5.5a) and (5.5b) as branching rules. The resulting branching rules are valid because, for every feasible solution $(\tau, \{\mathbf{R}_s\}_{s\in\mathcal{S}})$ in problem (5.4), there exist arrival-time vectors $\boldsymbol{a}_s \in \mathcal{X}(\mathbf{R}_s, [e, \ell]; \theta_s)$ for each $s \in \mathcal{S}$ that satisfy disjunctions (5.5).

$$[a_{si} \leq \beta + w_i/2 \; \forall s \in \mathcal{S}] \;\vee\; [a_{si} \geq \beta - w_i/2 \; \forall s \in \mathcal{S}] \qquad \forall \beta \in \mathbb{R},\; \forall i \in V_{\text{cont}} \tag{5.5a}$$

$$[a_{si} \leq \overline{y}_{ib} \; \forall s \in \mathcal{S}] \;\vee\; \left[a_{si} \geq \underline{y}_{i,b+1} \; \forall s \in \mathcal{S}\right] \qquad \forall b \in \{1, \ldots, N_i - 1\},\; \forall i \in V_{\text{disc}}. \tag{5.5b}$$

Conversely, if there exist route sets $\mathbf{R}_s \in \mathcal{R}([e, \ell]; \theta_s)$ and arrival-time vectors $\boldsymbol{a}_s \in \mathcal{X}(\mathbf{R}_s, [e, \ell]; \theta_s)$ for each $s \in \mathcal{S}$ satisfying disjunctions (5.5), then there exists a time window assignment $\tau \in TW$ such that $(\tau, \{\mathbf{R}_s\}_{s\in\mathcal{S}})$ is feasible in problem (5.4). Specifically, a feasible time window assignment is

$$\tau_i^\star = \begin{cases} [y_i, y_i + w_i] \text{, where } y_i = \min\left\{\ell_i - w_i, \min\limits_{s\in\mathcal{S}} a_{si}\right\} & \text{if } i \in V_{\text{cont}} \\ \left[\underline{y}_{ib_i}, \overline{y}_{ib_i}\right] \text{, where } b_i = \operatorname*{arg\,min}\limits_{b\in\{1,\ldots,N_i\}} \left\{\overline{y}_{ib} : \overline{y}_{ib} \geq \max\limits_{s\in\mathcal{S}} a_{si}\right\} & \text{if } i \in V_{\text{disc}} \end{cases} \qquad \forall i \in V_C. \tag{5.6}$$

For given route sets $\mathbf{R}_s \in \mathcal{R}([e, \ell]; \theta_s)$, verifying the existence of arrival-time vectors $\boldsymbol{a}_s \in \mathcal{X}(\mathbf{R}_s, [e, \ell]; \theta_s)$, $s \in \mathcal{S}$, that satisfy disjunctions (5.5) is equivalent to verifying that

the optimal objective value δ of the following mixed-integer linear optimization problem is non-positive.

$$
\begin{aligned}
\underset{\delta,a,z,\overline{\mu},\underline{\mu}}{\text{minimize}} \quad & \delta \\
\text{subject to} \quad & \delta \in \mathbb{R}_+, \;\; \boldsymbol{a}_s \in \mathcal{X}(\mathbf{R}_s, \tau; \theta_s), \; s \in \mathcal{S} \\
& \left.\begin{aligned}
& \overline{\mu}_i, \underline{\mu}_i \in \mathbb{R}_+, \;\; z_{ib} \in \{0,1\}, \; b \in \{1,\dots,N_i\} \\
& \sum_{b=1}^{N_i} z_{ib} = 1 \\
& z_{ib} = 1 \;\; \Rightarrow \;\; \overline{\mu}_i \geq a_{si} - \overline{y}_{ib} \quad \forall (s,b) \in \mathcal{S} \times \{1,\dots,N_i\} \\
& z_{ib} = 1 \;\; \Rightarrow \;\; \underline{\mu}_i \geq \underline{y}_{ib} - a_{si} \quad \forall (s,b) \in \mathcal{S} \times \{1,\dots,N_i\}
\end{aligned}\right\} \quad \forall i \in V_{\text{disc}} \qquad (5.7) \\
& \delta \geq a_{s_1 i} - a_{s_2 i} - w_i \quad \forall (s_1, s_2) \in \mathcal{S} \times \mathcal{S} \qquad \forall i \in V_{\text{cont}} \\
& \delta \geq \sum_{i \in V_{\text{disc}}} \left(\overline{\mu}_i + \underline{\mu}_i \right).
\end{aligned}
$$

In this problem, δ records the minimum possible violation of disjunctions (5.5) across all feasible arrival-time vectors $\boldsymbol{a}_s \in \mathcal{X}(\mathbf{R}_s, [e,\ell]; \theta_s)$, $s \in \mathcal{S}$. The second-to-last constraint ensures that δ is at least as large as the maximum violation across all members of (5.5a), while the last constraint ensures that δ is at least as large as the sum of violations across all members of (5.5b). In the latter case, the binary variable z_{ib} indicates whether the b^{th} member of (5.5b) is minimally violated for given $i \in V_{\text{disc}}$. In other words, if $z_{ib} = 1$, then $\left[\underline{y}_{ib}, \overline{y}_{ib}\right]$ is the best time window for customer i, and $\overline{\mu}_i = \left[\max_{s \in \mathcal{S}} a_{si} - \overline{y}_{ib}\right]_+$ and $\underline{\mu}_i = \left[\underline{y}_{ib} - \min_{s \in \mathcal{S}} a_{si}\right]_+$ respectively record the arrival-time violations with respect to the start and end of this time window (refer to Figure 5.2). Here, $[\cdot]_+ = \max\{\cdot, 0\}$.

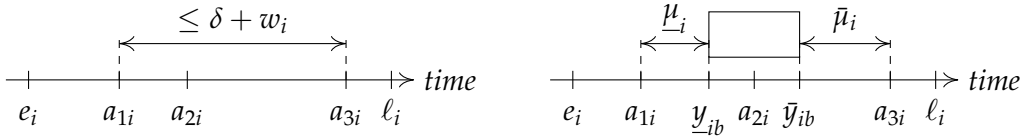


Figure 5.2: Decision variables in problem (5.7) for the continuous (left) and discrete (right) cases.

ALGORITHM. The above observations suggest that we can solve the TWAVRP without having to explicitly encode its time window assignments. In fact, any TWAVRP instance decomposes into its individual scenarios if we track, within each node of a branch-and-bound search tree, a vector of applicable time windows (one for each customer), which we shall denote by τ. The root node enforces only the exogenous time windows (see Step 1). Processing a node amounts to solving a set of VRPTW instances (one for each scenario) with time window constraints enforced by that node (see Step 3). It is important to remark that these VRPTW instances are uncoupled, and can thus be solved independently of each other. This is because none of the expressions within each disjunct in (5.5) (upon which our branching rules are based) link arrival-times from

different scenarios in the same inequality. The resulting optimal route sets $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$ are then used as inputs to the separation problem (5.7) (see Step 5). If the optimal objective value satisfies $\delta \leq 0$, then a new, improved time window assignment $\tau^\star$ is recorded, as per (5.6). Otherwise, the algorithm creates two new nodes with tightened time windows for customer $i^\star \in V_C$ (see Step 6). The overall algorithm is as follows.

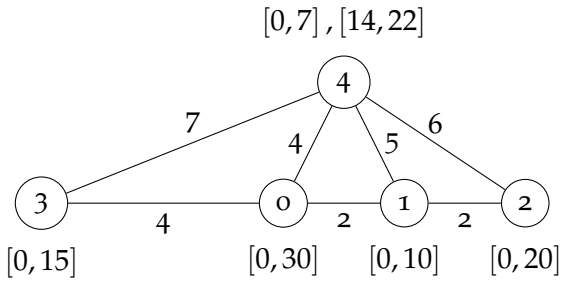
1. *Initialize.* Set root node $\tau^0 \leftarrow ([e_1, \ell_1], \ldots, [e_n, \ell_n])$, node queue $\mathcal{N} \leftarrow \{\tau^0\}$, upper bound $UB \leftarrow +\infty$ and optimal time window assignment $\tau^\star \leftarrow \emptyset$.
2. *Check convergence.* If $\mathcal{N} = \emptyset$, then stop: $\tau^\star$ is the optimal time window assignment with (expected) cost UB. Otherwise, select a node τ from $\mathcal{N}$, and set $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\tau\}$.
3. *Process node.* For each $s \in \mathcal{S}$, solve $\texttt{VRPTW}(\tau; \theta_s)$; and let $\mathbf{R}_s$ denote an optimal solution.
4. *Fathom by bound.* If $\sum_{s\in\mathcal{S}} p_s c(\mathbf{R}_s) \geq UB$, then go to Step 2.
5. *Check feasibility.* Let $(\delta, a, z, \overline{\mu}, \underline{\mu})$ be an optimal solution of problem (5.7). If $\delta \leq 0$, then a new, improved time window assignment is found: set $\tau^\star$ as per (5.6), set $UB \leftarrow \sum_{s\in\mathcal{S}} p_s c(\mathbf{R}_s)$ and go to Step 2.
6. *Branch.* Instantiate two children nodes, τ^{L} and τ^{R}, from the parent node: $\tau^{\mathrm{L}} \leftarrow \tau$, $\tau^{\mathrm{R}} \leftarrow \tau$. If $\delta > \sum_{i\in V_{\mathrm{disc}}} (\overline{\mu}_i + \underline{\mu}_i)$, then do Step 6a; otherwise, do Step 6b:
   a) *Branch as per* (5.5a). Let $i^\star \in V_{\mathrm{cont}}$ be any customer for which $\delta = \max_{s\in\mathcal{S}} a_{si^\star} - \min_{s\in\mathcal{S}} a_{si^\star} - w_{i^\star}$, and let $\beta^\star = \left(\max_{s\in\mathcal{S}} a_{si^\star} + \min_{s\in\mathcal{S}} a_{si^\star}\right)/2$. Tighten the time window for $i^\star$ as follows: *(i)* $\tau^{\mathrm{L}}_{i^\star} \leftarrow \left[\min_{t\in\tau^{\mathrm{L}}_{i^\star}} t, \beta^\star + \frac{1}{2} w_{i^\star}\right]$, *(ii)* $\tau^{\mathrm{R}}_{i^\star} \leftarrow \left[\beta^\star - \frac{1}{2} w_{i^\star}, \max_{t\in\tau^{\mathrm{R}}_{i^\star}} t\right]$.
   b) *Branch as per* (5.5b). Let $i^\star \in V_{\mathrm{disc}}$ be any member of $\arg\max_{i\in V_{\mathrm{disc}}} \{\overline{\mu}_i + \underline{\mu}_i\}$. If $\overline{\mu}_{i^\star} \geq \underline{\mu}_{i^\star}$, let $b^\star = \sum_{b=1}^{N_i} b \mathbb{1}_{[z_{ib}=1]}$; else, let $b^\star = \sum_{b=1}^{N_i} (b-1) \mathbb{1}_{[z_{ib}=1]}$. Tighten the time window for $i^\star$ as follows: *(i)* $\tau^{\mathrm{L}}_{i^\star} \leftarrow \left[\min_{t\in\tau^{\mathrm{L}}_{i^\star}} t, \overline{y}_{i^\star b^\star}\right]$, *(ii)* $\tau^{\mathrm{R}}_{i^\star} \leftarrow \left[\underline{y}_{i^\star\, b^\star+1}, \max_{t\in\tau^{\mathrm{R}}_{i^\star}} t\right]$.

   Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{\tau^{\mathrm{L}}, \tau^{\mathrm{R}}\}$, and go to Step 2.

We remark that any node selection strategy can be used in Step 2 to guarantee convergence.

An illustration on a small example is now presented to aid understanding and give intuition about the algorithm. Consider the TWAVRP instance shown in Figure 5.3. This example features $n = 4$ customers, with $V_{\mathrm{cont}} = \{1, 2, 3\}$ and $V_{\mathrm{disc}} = \{4\}$. Only customer demands are uncertain and they are represented using $S = 2$ scenarios.

The search tree of our algorithm to solve the illustrative example of Figure 5.3 is shown in Figure 5.4. Each "rectangle" denotes a node of our search tree. Within each rectangle, for each scenario $s \in \{1, 2\}$, the optimal route set $\mathbf{R}_s$ is shown. The x-coordinate of each customer $i \in V_C$ denotes its arrival-time a_{si}, which is computed by solving the separation



$(n, S, Q) = (4, 2, 3)$

	p_s	q_{s1}	q_{s2}	q_{s3}	q_{s4}
$s = 1$	1/2	1	3	1	1
$s = 2$	1/2	3	1	1	1

$w_1 = w_2 = w_3 = 3$

Figure 5.3: Instance parameters for the illustrative example. The arc weights denote both travel times and costs, and they are such that the triangle inequality is satisfied. Note that the graph is assumed to be complete, with nodes 0–3 lying along a straight line and costs/times to travel along this line being cumulative (not all arcs are shown for convenience). All service times are zero. The depicted intervals denote exogenous time windows for nodes 1–3 and feasible ones for node 4.

problem (5.7) to optimality (note that there may be multiple optimal solutions in each case). Finally, on top of each rectangle, the time windows τ enforced by our algorithm and the objective value of the corresponding optimal route sets (equal to $\sum_{s\in\mathcal{S}} p_s c(\mathbf{R}_s)$) are shown. In the root node, the separation problem (5.7) certifies (in Step 5) that no valid time window assignment can be constructed from its optimal solution. In particular, if we focus on customer $3 \in V_{\text{cont}}$, then $a_{13} = 14$ and $a_{23} = 5$. These arrival times clearly do not fall within a time window of width $w_3 = 3$. Therefore, as per Step 6a, $\beta^\star = (14 + 5)/2 = 9.5$. The time window of customer 3 in the left child is tightened to $[e_3, \beta^\star + w_3/2] = [0, 11]$, while in the right child to $[\beta^\star - w_3/2, \ell_3] = [8, 15]$. Similarly, if we focus on customer $4 \in V_{\text{disc}}$ in this right child node, then $a_{14} = 9$ and $a_{24} = 22$. These arrival times also do not simultaneously satisfy either of the two candidate windows, $[0, 7]$ or $[14, 22]$. Therefore, a branch is made, as per Step 6b, tightening the time window of customer 4 in the left child to $[0, 7]$ and in the right one to $[14, 22]$.

We remark that, in a given node of our search tree (except the root node), one does not need to solve a VRPTW subproblem for every scenario as required by Step 3 of the algorithm, and the optimal VRPTW route sets for some of the scenarios can be directly transferred from the parent subproblems. In fact, after branching has occurred in Step 6 of the algorithm, at most S (out of a total of $2S$) VRPTW subproblems have to be solved across the two children nodes. This is because, by construction, any arrival time vectors corresponding to the optimal solution, $\mathbf{R}_s$, of a given scenario s cannot simultaneously violate both disjuncts of the applied branching disjunction (either (5.5a) or (5.5b)), although it may satisfy both disjuncts simultaneously. Therefore, as far as a given scenario s is concerned, a VRPTW subproblem needs to be solved only at most once across the two children nodes. To illustrate this, see Figure 5.4. After branching has occurred in the root node, $\mathbf{R}_1$ in the right child node is exactly the same as that in the parent, since it already satisfies the applied branching constraint $[a_3 \geq 8]$. For the same reason, $\mathbf{R}_2$ in the left child node is exactly the same as that in the parent. Once the

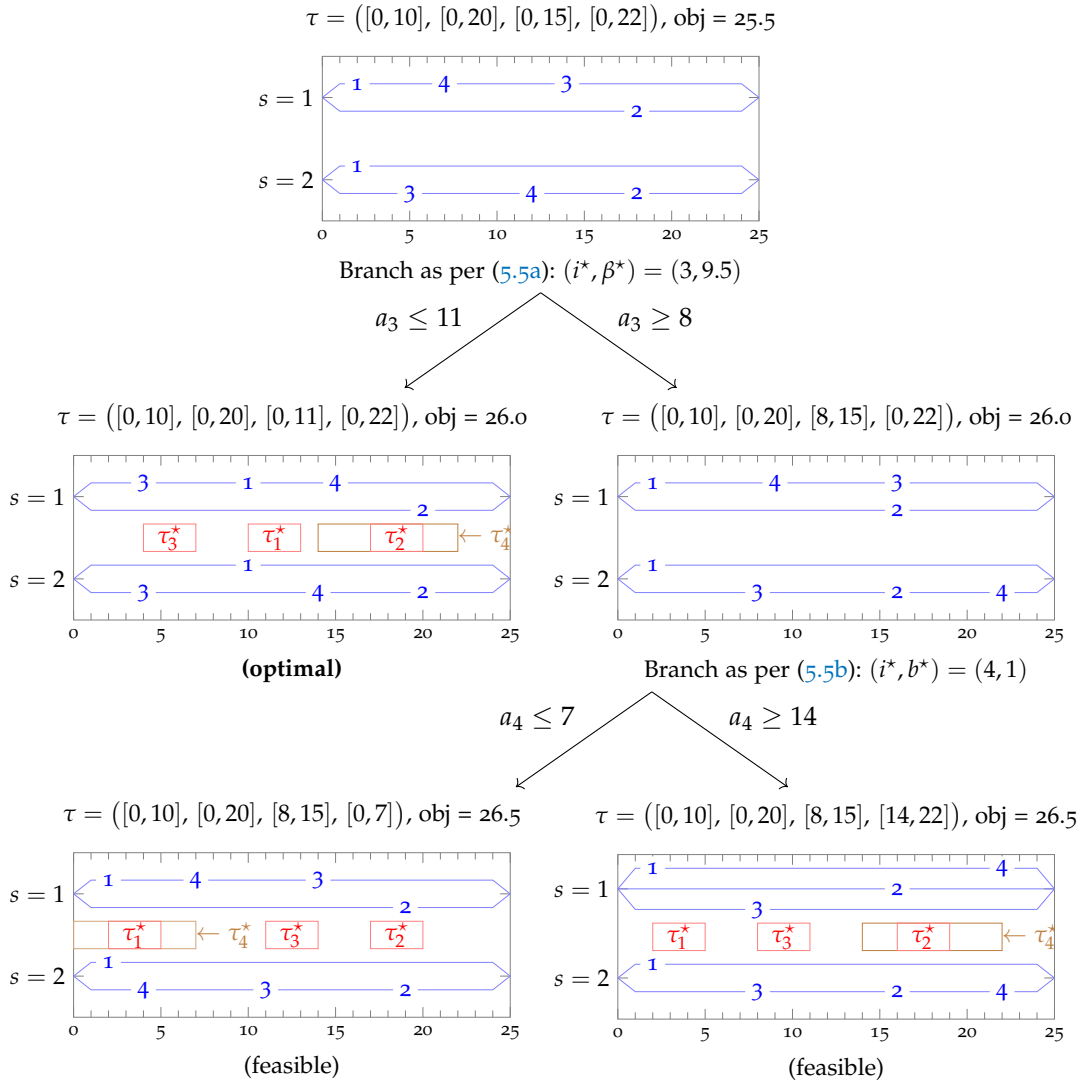


Figure 5.4: The search tree of our algorithm for the illustrative example of Figure 5.3.

branching rule has been established, whether a scenario-specific set of routes remains feasible (and hence, optimal) for the VRPTW instance of a child node can be inferred trivially by inspection, and hence, the corresponding instance need not be solved, as the routes can be copied over. For the VRPTW instances that indeed warrant a new route set to be computed, Section 5.5 describes methods for solving the corresponding subproblems.

5.4.2 *Path-based Disjunctions*

The algorithm described in the previous section converges in finite time (the argument is similar to that in Chapter 4). However, the branching Step 6 may not necessarily "cut off" the optimal VRPTW route set $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$ found in the parent node. Indeed, it is only guaranteed to cut off the arrival time vector $\{\boldsymbol{a}_s\}_{s\in\mathcal{S}}$ corresponding to $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$. More specifically, the optimal route set of the (right) child node τ^{R} may be exactly the same as that of its parent node τ. This is because the only difference between τ^{R} and its parent τ is that the former features a tighter earliest start time for customer $i^\star \in V_C$. It is possible for the optimal route set of τ to be unchanged, if the tighter earliest start time constraint can be satisfied by simply *allowing the vehicle to wait longer at* $i^\star$. To see this, consider again the example of Figure 5.3. If the exogenous time window of customer 2 is increased by one unit to $[0, 21]$, then the optimal route set $\mathbf{R}_2$ in the root node of the algorithm (see Figure 5.4) will be exactly the same as that in its right child node since the vehicle visiting customer 3 will simply wait longer until it satisfies the branching constraint, $a_3 \geq 8$. Along with the fact that $\mathbf{R}_1$ is also the same (refer to the discussion in the previous section), this means that the optimal VRPTW route set $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$ in the root node has not changed in its right child node. The impact of this is non-improving lower bounds: the objective value of the node τ^{R} will be exactly the same as that of its parent, leading to slow convergence and poor numerical performance. This observation motivates us to investigate branching rules which are guaranteed to cut off the parent route set.

Our motivation for the new class of disjunctions comes from the *path precedence inequalities* proposed in [94] for the continuous TWAVRP and the *inconsistent path elimination constraints* proposed in Chapter 4 for the ConTSP. Consider a feasible solution $(\tau, \{\mathbf{R}_s\}_{s\in\mathcal{S}})$ to problem (5.4). Suppose that there is a vehicle route in the solution $\mathbf{R}_{s_1}$ of scenario $s_1 \in \mathcal{S}$ in which customer $i \in V_C$ is visited before customer $j \in V_C \setminus \{i\}$, and that there is a vehicle route in the solution $\mathbf{R}_{s_2}$ of scenario $s_2 \in \mathcal{S} \setminus \{s_1\}$ in which j is visited before i. Since both i and j are visited within their respective time windows τ_i and τ_j in both scenarios, it must be the case that the sum of the travel times from i to j in scenario s_1 and from j to i in scenario s_2 is at most the sum of the widths of their time windows, $w_i + w_j$, as shown in Figure 5.5. Consequently, if this condition is not satisfied by a route set $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$ for all possible pairs (i, j), then there cannot exist a feasible time window assignment $\tau \in TW$ such that $(\tau, \{\mathbf{R}_s\}_{s\in\mathcal{S}})$ is feasible in problem (5.4).

To construct valid disjunctions based on the above observation, we first introduce some notation. Let $\pi = (v_1, \ldots, v_p)$ denote a directed $v_1 - v_p$ path in graph G that is formed by the arcs in the set $\{(v_i, v_{i+1}) : i = 1, \ldots, p-1\}$, where $(v_i, v_{i+1}) \in A$ for all $i = 1, \ldots, p-1$. We shall only consider paths which are open and simple, i.e., $p > 1$ and $v_i \neq v_j$ for $i \neq j$. For a given realization of the travel and service times, the travel time along π is defined to be $t(\pi) = \sum_{i=1}^{p-1} \left(t_{v_i v_{i+1}} + u_{v_i}\right)$, where we define $u_0 = 0$. Note that as per this definition, the travel time along a path does not include any waiting time that might potentially be incurred at its nodes. Finally, a route set $\mathbf{R} = (R_1, \ldots, R_m)$, where $R_k = (R_{k,1}, \ldots, R_{k,n_k})$

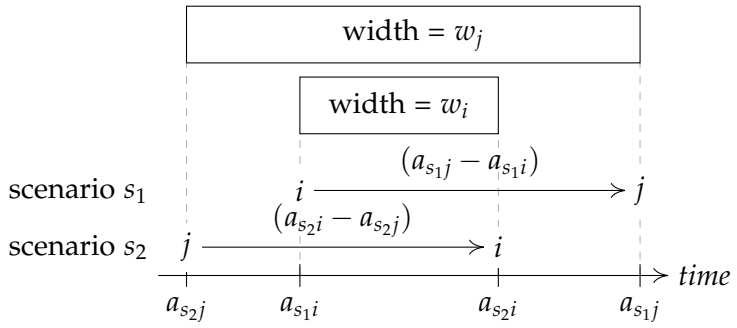


Figure 5.5: Motivation for the path-based disjunctions. The above shown sub-paths must satisfy $(a_{s_2 i} - a_{s_2 j}) + (a_{s_1 j} - a_{s_1 i}) \leq w_i + w_j$, if they are to be part of a feasible solution.

for $k = 1, \ldots, m$, is said to contain π, if π appears as a sub-path in any of its routes, i.e., if for some k, we have $R_k = (R_{k,1}, \ldots, R_{k,l} = v_1, R_{k,l+1} = v_2, \ldots, R_{k,l+p-1} = v_p, \ldots, R_{k,n_k})$. The key result of this section is that every feasible solution $(\tau, \{\mathbf{R}_s\}_{s \in \mathcal{S}})$ to problem (5.4) satisfies the following disjunctions:

$$\left[\begin{array}{l} \mathbf{R}_s \text{ does not contain any } i - j \\ \text{path with travel time} \geq d_1 \end{array} \quad \forall s \in \mathcal{S}\right] \vee \left[\begin{array}{l} \mathbf{R}_s \text{ does not contain any } j - i \\ \text{path with travel time} \geq d_2 \end{array} \quad \forall s \in \mathcal{S}\right]$$
$$\forall (d_1, d_2) \in \mathbb{R}^2 : d_1 + d_2 > w_i + w_j, \quad \forall (i, j) \in V_C \times V_C : i \neq j, \tag{5.8}$$

where w_k for any $k \in V_{\text{disc}}$ is defined to be $w_k = \max_{b \in \{1,\ldots,N_k\}} \{\overline{y}_{kb} - \underline{y}_{kb}\}$. However, the converse is not true. To see this, consider the following counter-example.

- $(n, S, Q) = (4, 2, 3)$. $V_{\text{cont}} = V_C$ and $w_i = 1$ for all $i \in V_C$.
- $[e, \ell] = ([0, 6], [0, 6], [3, 4], [4, 5])$. Also, $[e_0, \ell_0] = [0, 10]$.
- $G = (V, A)$ is complete. $c_{ij} = t_{ij} = 1$ for all $(i, j) \in A$ and $u_i = 0$ for all $i \in V_C$.
- Demand is uncertain. $(q_{s1}, q_{s2}, q_{s3}, q_{s4}) = (1, 1, 1, 3)$ for $s = 1$ and $(1, 1, 3, 1)$ for $s = 2$.

Consider the route sets $\{\mathbf{R}_s\}_{s=1,2}$ shown in Figure 5.6. The x-coordinates correspond to arrival times. This solution satisfies all path-based disjunctions (5.8). However, it is not a feasible TWAVRP solution since there are no valid time window assignments $\tau \in TW$ for customers 1 and 2. In contrast, observe that this solution does indeed violate the (necessary and sufficient) time window-based disjunctions (5.5a) corresponding to $(i, \beta) = (1, 4)$ as well as $(i, \beta) = (2, 4)$.

The above observations suggest that we can use the path-based disjunctions (5.8) as the basis of a branching rule in addition to the time window-based disjunctions (5.5). The corresponding branching constraints, i.e., constraints within each individual disjunct of (5.8), are equivalent to *path elimination constraints* (e.g., see [17]). Consequently, the subproblems to be solved in Step 3 of the algorithm are VRPTW instances with additional

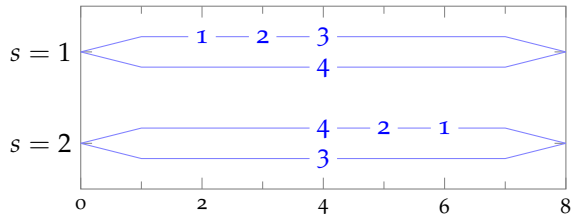


Figure 5.6: Counter-example to show that the disjunctions (5.8) are not sufficient for the TWAVRP.

path elimination constraints. To describe these subproblems formally, if $\mathcal{F}$ is a finite collection of triples $(i, j, d) \in V_C \times V_C \times \mathbb{R}$, such that $i \neq j$, then we let each member of $\mathcal{F}$ represent a family of *forbidden paths*. Specifically, the member $(i, j, d) \in \mathcal{F}$ represents the set of all $i - j$ paths with travel time greater than or equal to d. Given a collection $\mathcal{F}$ of representative forbidden paths, time windows τ and operational parameters θ, we define $\texttt{VRPTWFP}(\tau, \mathcal{F}; \theta)$ to be the optimal value of the following optimization problem:

$$\begin{aligned} \underset{\mathbf{R}}{\text{minimize}} \quad & c(\mathbf{R}) \\ \text{subject to} \quad & \mathbf{R} \in \mathcal{R}(\tau; \theta) \\ & \mathbf{R} \text{ does not contain any } i\text{-}j \text{ path } \pi \text{ with } t(\pi) \geq d \quad \forall (i, j, d) \in \mathcal{F}. \end{aligned} \tag{$\texttt{VRPTWFP}(\tau, \mathcal{F}; \theta)$}$$

Note that $\texttt{VRPTWFP}(\tau, \emptyset; \theta) = \texttt{VRPTW}(\tau; \theta)$ so our definition is consistent with the one in Section 5.3.1.

Before we can incorporate the path-based disjunctions (5.8) as branching rules in our algorithm, we also need a separation algorithm, which will take as input route sets $\{\mathbf{R}_s\}_{s \in \mathcal{S}}$, and return either a violated member of (5.8) or a certificate that all of its members are satisfied. We compute the following quantities, where we assume $(i, j) \in V_C \times V_C$, such that $i \neq j$, is a given pair of customers.

- $\mathcal{S}_{ij}$: scenarios containing an $i - j$ path; that is, $\mathcal{S}_{ij} = \{s \in \mathcal{S} : \mathbf{R}_s \text{ contains an } i - j \text{ path}\}$.
- d_{ij}^s: travel time of $i - j$ path in $\mathbf{R}_s$, where $s \in \mathcal{S}_{ij}$.
- ν_{ij}: no. of violating scenario pairs; that is, $\nu_{ij} = \left| \left\{ (s_1, s_2) \in \mathcal{S}_{ij} \times \mathcal{S}_{ji} : d_{ij}^{s_1} + d_{ji}^{s_2} > w_i + w_j \right\} \right|$.
- Δ_{ij}: minimum value of sum of path travel times (across violating scenario pairs); that is, $\Delta_{ij} = \inf_{(s_1, s_2) \in \mathcal{S}_{ij} \times \mathcal{S}_{ji}} \left\{ d_{ij}^{s_1} + d_{ji}^{s_2} : d_{ij}^{s_1} + d_{ji}^{s_2} > w_i + w_j \right\}$.

We are now in a position to incorporate the path-based-disjunctions (5.8) in our algorithm. To do so, we store the set of forbidden paths $\mathcal{F}$ as part of a node's characteristic data (along with τ) at the time of node creation (initialization and branching steps), and we use this set as input to the VRPTW subproblem at the time of node processing. We

update $\mathcal{F}$ in the branching step based on a ranked list $\mathcal{L}$ of pairs (i,j) for which the corresponding path-based disjunctions (5.8) can be used as branching rules. In all, the following modifications are made to the main algorithm.

1′ *Initialize.* Set root node $\tau^0 \leftarrow ([e_1,\ell_1],\ldots,[e_n,\ell_n])$, $\mathcal{F}^0 \leftarrow \emptyset$, node queue $\mathcal{N} \leftarrow \{(\tau^0,\mathcal{F}^0)\}$, upper bound $UB \leftarrow +\infty$ and optimal time window assignment $\tau^\star \leftarrow \emptyset$.

3′ *Process node.* For each $s \in \mathcal{S}$, solve $\texttt{VRPTWFP}(\tau,\mathcal{F};\theta_s)$; and let $\mathbf{R}_s$ denote its optimal solution.

6′ *Branch.* Instantiate two children nodes from $(\tau,\mathcal{F})$: $(\tau^{\mathrm{L}},\mathcal{F}^{\mathrm{L}}) \leftarrow (\tau,\mathcal{F})$, $(\tau^{\mathrm{R}},\mathcal{F}^{\mathrm{R}}) \leftarrow (\tau,\mathcal{F})$. Set $\mathcal{L} \leftarrow \emptyset$. For each pair $(i,j) \in V_{\mathrm{C}} \times V_{\mathrm{C}}$, such that $i \neq j$ and $v_{ij} \geq 1$, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(i,j)\}$.
If $\mathcal{L} \neq \emptyset$, do Step 6′c; else if $\delta > \sum_{i \in V_{\mathrm{disc}}} (\mu_i^+ + \mu_i^-)$, do Step 6a; else, do Step 6b:

(c) Sort $\mathcal{L}$ in decreasing order of v (breaking ties in increasing order of Δ). Let (i,j) be the first element of $\mathcal{L}$ and let $(s_1,s_2) = \arg\min_{(s,s') \in \mathcal{S}_{ij} \times \mathcal{S}_{ji}} \left\{ d_{ij}^s + d_{ji}^{s'} : d_{ij}^s + d_{ji}^{s'} > w_i + w_j \right\}$. Set $d_1^\star$ and $d_2^\star$ as follows: if $d_{ij}^{s_1} \leq d_{ji}^{s_2}$, set $d_1^\star \leftarrow d_{ij}^{s_1}$, $d_2^\star \leftarrow w_i + w_j - d_1^\star + \varepsilon$; otherwise, set $d_2^\star \leftarrow d_{ji}^{s_2}$, $d_1^\star \leftarrow w_i + w_j - d_2^\star + \varepsilon$; here, ε is a small positive number. Set $i^\star \leftarrow i$, $j^\star \leftarrow j$. Add path elimination constraints: *(i)* $\mathcal{F}^{\mathrm{L}} \leftarrow \mathcal{F}^{\mathrm{L}} \cup \{(i^\star,j^\star,d_1^\star)\}$, *(ii)* $\mathcal{F}^{\mathrm{R}} \leftarrow \mathcal{F}^{\mathrm{R}} \cup \{(j^\star,i^\star,d_2^\star)\}$.

Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\tau^{\mathrm{L}},\mathcal{F}^{\mathrm{L}}),(\tau^{\mathrm{R}},\mathcal{F}^{\mathrm{R}})\}$, and go to Step 2.

The modified algorithm gives preference to the branching Step 6′c over Steps 6a and 6b, because unlike the latter, the former would necessarily cut off the VRPTW route set of the parent node in at least one scenario (in both children nodes). However, as discussed earlier, the path-based disjunctions (5.8) (upon which branching rule 6′c is based) are only necessary but not sufficient. This is in contrast to the time window-based disjunctions (5.5) (upon which the branching rules 6a and 6b are based), which are both necessary and sufficient.

Figure 5.7 shows the effect that the new branching rules have on the branch-and-bound search tree for the case of our illustrative example from Figure 5.3. Note how, in the root node, instead of branching via the time window-based disjunctions (5.5), the modified branching Step 6′ certifies that it is impossible to have both $3-4$ and $4-3$ paths in different scenario route sets, and branches using the disjunction (5.8) instead. Observe that the search tree is much smaller than in the case of Figure 5.4. Our numerical experiments confirm that this is generally true, i.e., that incorporating the path-based disjunctions (5.8) results in fewer nodes being explored (see Section 5.6.3).

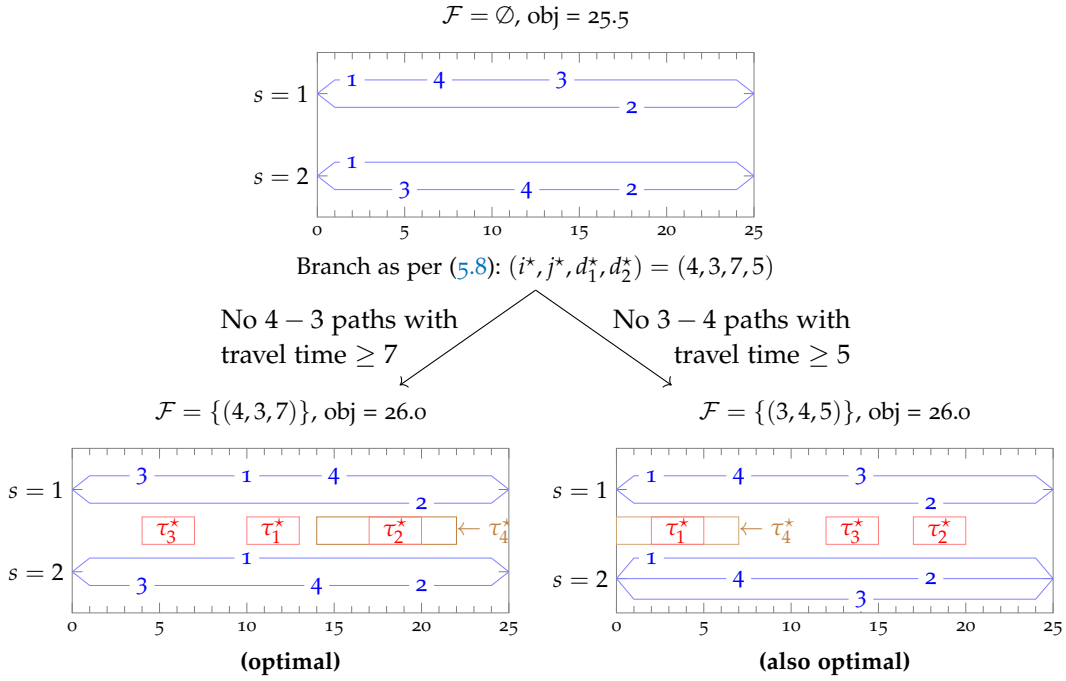


Figure 5.7: The search tree of our algorithm (utilizing path-based disjunctions) for the illustrative example of Figure 5.3.

5.4.3 *Generating Upper Bounds*

A small value of the global upper bound UB can significantly speed up the solution process by fathoming more nodes of the search tree using the fathoming Step 4. The algorithm presented in Section 5.4.1 updates UB in Step 5 only. However, it is possible to update UB more frequently by generating candidate (feasible) TWAVRP solutions using the route sets $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$ obtained in the node processing Step 3. The basic idea is to select some scenario and use its solution as a "template" [140], assigning the time windows based on the arrival times in this template solution. Specifically, for every scenario $s^\star \in \mathcal{S}$ in which a new route set $\mathbf{R}_{s^\star}$ is computed in Step 3 of the algorithm, we can attempt to generate a feasible TWAVRP solution $(\tau^h, \{\mathbf{R}_s^h\}_{s\in\mathcal{S}})$ using the following procedure.

1. *Initialize arrival times.* Let $\mathbf{R} \leftarrow \mathbf{R}_{s^\star}$ and $\theta \leftarrow \theta_{s^\star}$. Let $\boldsymbol{a} \in \mathcal{X}(\mathbf{R}, [e, \ell]; \theta)$ be the vector of arrival times with minimum cumulative waiting time. That is, we recursively define $a_{R_{k,0}} = e_0$ and $a_{R_{k,l+1}} = \max\{e_{R_{k,l+1}}, a_{R_{k,l}} + t_{R_{k,l},R_{k,l+1}}\}$ for all $l \in \{1, \ldots, n_k - 1\}$ and $k \in \{1, \ldots, m\}$. For $i \in V_C$ such that $q_i = 0$ (that is, $i \neq R_{k,l}$ for any k, l), we define $a_i = e_i$.

2. *Assign time windows.* Let τ^h be defined as follows.

$$\tau_i^h = \begin{cases} [x_i, x_i + w_i] \text{, where } x_i = \begin{cases} e_i & \text{if } a_i - w_i/2 \leq e_i \\ \ell_i - w_i & \text{if } a_i + w_i/2 \geq \ell_i \\ a_i - w_i/2 & \text{otherwise} \end{cases} & \text{if } i \in V_{\text{cont}} \\ \left[x_{i,b_i}, y_{i,b_i}\right] \text{, where } b_i \in \underset{b \in \{1,\ldots,N_i\}}{\arg\min} \left\{ \min_{\omega \in [x_{i,b}, y_{i,b}]} |a_i - \omega| \right\} & \text{if } i \in V_{\text{disc}} \end{cases} \qquad \forall i \in V_C.$$

3. *Compute upper bound.* For each $s \in \mathcal{S}$, let $\mathbf{R}_s^h$ be a (possibly suboptimal) solution of $\texttt{VRPTW}(\tau^h; \theta_s)$. Let $ub^h \leftarrow \sum_{s \in \mathcal{S}} p_s c(\mathbf{R}_s^h)$. If $ub^h \leq UB$, set $UB \leftarrow ub^h$ and $\tau^\star \leftarrow \tau^h$.

We remark that it is not necessary to solve the VRPTW instances to optimality in Step 3 above, since the generated upper bounds ub^h are guaranteed to still be valid. Consequently, we can utilize any (possibly heuristic) VRPTW solver to quickly compute candidate upper bounds. For example, the computational results reported in Section 5.6 were obtained by implementing Solomon's sequential insertion construction heuristic "I1" [238] in combination with a local search procedure [65], which used the *Relocate*, *2-opt*, *2-opt** and *Or-opt* moves within a deterministic Variable Neighborhood Descent algorithm (e.g., see [64]).

5.4.4 *Modification as a Heuristic Algorithm*

The exact algorithm of Section 5.4.1 can be readily modified as a heuristic algorithm. Indeed, if one uses a heuristic VRPTW solver (e.g., one based on a metaheuristic) in place of an exact one in the node processing Step 3, then the time window assignment $\tau^\star$ determined by the algorithm is still guaranteed to be feasible, although not necessarily optimal. Overall TWAVRP optimality can no longer be guaranteed, since the fathoming Step 4 may incorrectly prune a node whose descendant contains the optimal time window assignment. Nevertheless, this modification as a heuristic is particularly suited from a practical viewpoint, as it allows one to utilize any available VRPTW heuristic solver "out of the box."

5.5 EXACT SOLUTION OF VRPTW SUBPROBLEMS

Various exact solution schemes have been proposed for the VRPTW over the last several decades. These include algorithms based on branch-and-cut, Lagrangean relaxation and column generation, among others. We refer the reader to [102] for a recent survey. The most successful of these are *branch-price-and-cut* algorithms, which correspond to branch-and-bound algorithms in which the bounds are obtained by solving linear relaxations of a *set partitioning* model by column generation, and are further strengthened by generating cutting planes.

In order to solve VRPTW instances in Steps 3 and 3′ of the main algorithm, we implemented the branch-price-and-cut algorithm described in [210], as well as procedures to warm start the latter exact method in the context of our algorithm. Our implementation incorporates several elements of the algorithm described in [210], including *ng*-routes, bidirectional labeling, variable fixing, route enumeration, and limited-memory subset row cuts. In what follows, we highlight only the most important of these ingredients; for details, we refer the reader to [210].

5.5.1 *Branch-Price-and-Cut Implementation*

For ease of notation, we shall drop the subscript s referencing a scenario and assume that all operational parameters θ are fixed to certain given values. We shall also assume that the time windows have been fixed at τ and that the set of forbidden paths is given to be $\mathcal{F}$. Note that we only describe the solution approach for $\texttt{VRPTWFP}(\tau, \mathcal{F}; \theta)$; the approach for $\texttt{VRPTW}(\tau; \theta)$ is obtained by simply setting $\mathcal{F} = \emptyset$. We remark that, before we call the exact solution method, we modify the VRPTW instance to obtain an equivalent one by tightening the time windows τ and reducing the arc set A using the preprocessing routines described in [17, 164]. In addition to this, it is possible to further reduce the arc set A using members of $\mathcal{F}$. Indeed, if for some $(i, j, d) \in \mathcal{F}$, the shortest travel time from i to j in graph G exceeds d, then we can remove the arc (i, j) from A.

In the following, $\mathcal{P}_{ij}$ denotes the set of all $i - j$ paths in graph G (after preprocessing), where $(i, j) \in V_C \times V_C$, such that $i \neq j$. We use Ω to denote the set of all (elementary) vehicle routes that are feasible with respect to capacity and time window constraints. For a given route $r \in \Omega$, λ_{ir} denotes the number of times customer $i \in V_C$ is visited in route r, η_{ijr} denotes the number of times arc $(i, j) \in A$ is traversed by route r, while c_r denotes its cost, i.e., $c_r \equiv c(r)$. The set partitioning model is described in the following. In this model, x_r is a binary path-flow variable that encodes whether route $r \in \Omega$ is part of the optimal route set.

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \sum_{r \in \Omega} c_r x_r \\
\text{subject to} \quad & x_r \in \{0, 1\}, \quad \forall r \in \Omega, \\
& \sum_{r \in \Omega} \lambda_{ir} x_r = 1, \quad \forall i \in V_C, \\
& \sum_{r \in \Omega} \sum_{(i', j') \in \pi} \eta_{i'j'r} x_r \leq |\pi| - 1, \quad \forall \pi \in \mathcal{P}_{ij} : t(\pi) \geq d, \ \ \forall (i, j, d) \in \mathcal{F}.
\end{aligned}
\tag{5.9}
$$

In the above, the last set of inequalities are *infeasible path elimination constraints* (e.g., see [17, 164]) that forbid the occurrence of $i - j$ paths with travel time greater than or equal to d. In our implementation, we replace the subscript of the innermost summation

with $(i', j') \in \text{tr.cl.}(\pi)$, where $\text{tr.cl.}(\pi)$ denotes the transitive closure of π.[3] This so-called *tournament form* of the inequality is stronger than the version presented above, see [17] for a proof. Furthermore, it is also well known that one can relax the feasible space of the above set partitioning model by including non-elementary vehicle routes in Ω without sacrificing optimality. In our implementation, we replace Ω with the set of so-called *ng*-routes $\Omega^{ng} \supseteq \Omega$, which are not necessarily elementary [30].

We now describe the branch-price-and-cut algorithm to solve the set partitioning model over Ω^{ng}. The root node, which solves the linear relaxation of this set partitioning model, is initialized with a subset of Ω^{ng} (single-customer vehicle routes) but no infeasible path elimination constraints. A pricing subproblem is used to generate other members of Ω^{ng} (also referred to as columns), as necessary. After column generation has converged, if the gap between the current node lower bound and global upper bound is sufficiently small ($\leq 1\%$ in our implementation), we employ *route enumeration* to generate all feasible vehicle routes with reduced costs smaller than this gap [28]. [28] have shown that this subset must contain the routes of all optimal solutions. Therefore, we can solve the resulting "reduced" set partitioning model by using a standard integer programming solver. We remark that this is done only if the number of generated routes is less than a threshold, which we set to 3×10^6, as suggested in [210]. On the other hand, if the current node gap is large or if route enumeration generated too many routes, then we attempt to separate the infeasible path elimination constraints (as well as other valid inequalities) in order to tighten the linear relaxation. The above process is iterated until we cannot generate any more columns or inequalities. At this stage, if the current node solution is fractional, we create additional children nodes by branching on the number of used vehicles or by branching on edges/arcs.

PRICING SUBPROBLEM. The pricing subproblem is a *shortest path problem with resource constraints*, with customer demands and arc travel times considered as resources constrained by the vehicle capacity and time windows, respectively. We utilize the dynamic programming algorithm described in [210] to solve this problem. In order to speed up the solution of the pricing subproblem, we apply various techniques including bidirectional labeling and variable fixing (based on reduced costs). In addition, we implemented the bucket pruning heuristic (e.g., see [119]) to find candidate columns and use the dynamic programming algorithm only if the former fails to generate columns.

ROUTE ENUMERATION. We use the dynamic programming algorithm of [210] with a modification to account for the presence of the infeasible path elimination constraints. In particular, we consider two different "partial routes" that visit the same set of customers (but possibly in different sequences) to be undominated irrespectively of their resource consumptions, and we do not perform any associated dominance checks. This prevents

3 If $\pi = (v_1, \ldots, v_p)$ denotes an elementary path, then its transitive closure is the set of arcs (v_k, v_l) such that v_l can be reached from v_k using only arcs in π, i.e, $\text{tr.cl.}(\pi) = \{(v_k, v_l) \in A : (k, l) \in \{1, \ldots, p\} \times \{1, \ldots, p\}, k < l\}$.

incorrectly "pruning" a vehicle route that satisfies the infeasible path elimination constraints in favor of one that does not. Among all enumerated routes, we only consider those which are elementary and satisfy all infeasible path elimination constraints (which can be done in $\mathcal{O}(|\mathcal{F}|)$ time per route) to include in the final "reduced" set partitioning model.

CUTTING PLANES. We use the tournament form of the infeasible path elimination constraints as "necessary cuts" (a violating member is guaranteed to be separated), and the *extended capacity cuts* [213] and *limited-node-memory subset row cuts* [210] as "strengthening cuts" (a violating member may not necessarily be separated). The separation algorithms for these cuts are described next.

- The infeasible path elimination constraints are separated by utilizing the polynomial-time path-growing scheme of [17]. Specifically, suppose $G_{\bar{x}} = (V, A_{\bar{x}})$ is the so-called support graph of the current linear programming solution $\bar{x}$, where $A_{\bar{x}} = \left\{(i', j') \in A : \sum_{r \in \Omega^{\text{ng}}} \eta_{i'j'r} \bar{x}_r > 0\right\}$. Then, for every $(i, j, d) \in \mathcal{F}$, the scheme of [17] is used to obtain the set of all $i - j$ paths π in $G_{\bar{x}}$, which satisfy $\sum_{r \in \Omega^{\text{ng}}} \sum_{(i',j') \in \text{tr.cl.}(\pi)} \eta_{i'j'r} \bar{x}_r > |\pi| - 1$. Amongst all such paths, we choose the ones for which the travel time $t(\pi)$ is greater than d and add the corresponding tournament form of the infeasible path elimination constraints to the current linear relaxation.
- The extended capacity cuts are separated by first using the CVRPSEP package [193] to separate the so-called *rounded capacity inequalities* and then lifting these, as described in [213].
- The limited-node-memory subset row cuts are separated by first identifying all subset row cuts violated by node sets with cardinality up to 5 and then identifying their so-called "node memory sets," as described in [210].

We remark that, since the infeasible path elimination inequalities are defined over arcs, they are "robust" and affect the pricing subproblem only through a corresponding term in the arc cost, i.e., their dual value. In other words, the addition of these inequalities does not affect the complexity of the pricing subproblem. In contrast, the extended capacity and limited-node-memory subset row cuts are "non-robust" as their addition increases the complexity of the pricing subproblem [210].

5.5.2 *Warm Starting*

Initializing the branch-price-and-cut algorithm described in the previous section with a feasible set of columns can speed up the convergence of its column generation process, leading to small computation times. In addition to this, providing a valid initial upper bound on the optimal objective value can also significantly speed up the search, both in the context of route enumeration, where it can result in fewer routes being enumerated,

as well as branch-and-bound, where more parts of the search tree can be fathomed early in the process. In this section, we describe warm starting procedures that can be employed in the context of our algorithm of Section 5.4.

Consider a parent node $(\tau, \mathcal{F})$ (already processed) with optimal solution $\{\mathbf{R}_s\}_{s\in\mathcal{S}}$ as obtained in Step 3. Also, consider one of its child nodes $(\tau^h, \mathcal{F}^h)$, where $h \in \{\mathrm{L}, \mathrm{R}\}$, created in the branching Step 6′. Finally, consider a scenario $s \in \mathcal{S}$ such that the corresponding optimal route set in the parent node, $\mathbf{R}_s$, is not feasible for the child node $(\tau^h, \mathcal{F}^h)$. This means that $\mathbf{R}_s$ cannot be directly transferred to the latter, and warm starting is sought to aid the search towards the optimal one.

GENERATING AN INITIAL SET OF COLUMNS. Let Ω_s^{ng} be the set of columns that were generated during the branch-price-and-cut process in scenario s of the parent node $(\tau, \mathcal{F})$. Since the child node $(\tau^h, \mathcal{F}^h)$ differs from its parent in exactly one constraint, it is likely that several members of Ω_s^{ng} are also feasible in the set partitioning model of the child's scenario$-s$ VRPTW subproblem. Therefore, we can simply loop through the members of Ω_s^{ng} and filter out all infeasible columns to generate the initial linear relaxation in the branch-price-and-cut algorithm.

GENERATING AN INITIAL UPPER BOUND. The following procedure computes a valid upper bound, ub_s^h, on the cost of the child's scenario$-s$ VRPTW subproblem. The procedure attempts to "repair" the scenario$-s$ optimal route set of the parent node and generate one that is feasible for the child.

1. Set $ub_s^h \leftarrow UB - \sum_{s'\in\mathcal{S}\setminus\{s\}} c(\mathbf{R}_{s'})$, where UB is the currently applicable upper bound from the algorithm of Section 5.4; and set $\mathbf{R}' \leftarrow \mathbf{R}_s$.
2. Let $H \subseteq V_C$ be defined as follows: if $(\tau^h, \mathcal{F}^h)$ was created using branching Steps 6a or 6b, then $H = \{i^\star\}$; otherwise $H = \{i^\star, j^\star\}$.
3. For each $i \in H$:
   a) Remove customer i from its current position in $\mathbf{R}'$ and insert it into a new vehicle route.
   b) Apply local search on $\mathbf{R}'$, ensuring that each accepted move satisfies all time window and path elimination constraints in $(\tau^h, \mathcal{F}^h)$.
   c) If $c(\mathbf{R}') < ub_s^h$, set $ub_s^h \leftarrow c(\mathbf{R}')$.

In our implementation of local search, we considered the *Relocate*, *2-opt*, *2-opt** and *Or-opt* moves within a deterministic Variable Neighborhood Descent algorithm (e.g., see [64]). We remark that the solution of (up to S) VRPTW instances in Step 3 of the algorithm can be easily parallelized. Alternatively, one can do this serially on a single CPU thread (e.g., by starting with the lowest indexed scenario). In the former setting, no information can be exchanged among the VRPTW instances. In the latter setting, however, one can

capitalize on instances that have already been solved so as to obtain improved upper bounds in Step 1 of the above procedure as follows:

$$ub_s^h \leftarrow UB - \sum_{s' \in \mathcal{S}: s' < s} c(\mathbf{R}_{s'}^h) - \sum_{s' \in \mathcal{S}: s' > s} c(\mathbf{R}_{s'}),$$

where $\mathbf{R}_{s'}^h$ is the just computed optimal solution in scenario s' of the node under consideration.

5.6 COMPUTATIONAL RESULTS

This section presents computational results obtained by our algorithm on benchmark instances from the literature. Specifically, in Section 5.6.1, we present the characteristics of the test instances; in Section 5.6.2, we present a summary of the numerical performance of our algorithm and compare it to existing solution methods; in Section 5.6.3, we present detailed tables of results outlining the performance of each component of our algorithm; and, finally in Section 5.6.4, we present results of a parallel implementation on instances containing a large number of scenarios.

The algorithm was coded in C++ and the runs were conducted on an Intel Xeon E5-2687W 3.1 GHz processor with 4 GB of available RAM. The nodes in Step 2 of the algorithm were selected using a simple depth-first rule that backtracked whenever the gap between the objective value of the current node and the current upper bound exceeded 50% of the gap between the global lower and current upper bound. All subordinate linear and mixed-integer linear programs were solved using default settings of the IBM ILOG CPLEX Optimizer 12.7. Finally, except for the results presented in Section 5.6.4, all runs were restricted to a single CPU thread. This facilitates a fair comparison with existing algorithms from the literature in Section 5.6.2 and between different configurations of our algorithm in Section 5.6.3. The results presented in Section 5.6.4 were obtained with OpenMP by using up to $\min\{S, 10\}$ threads in parallel, where S is the number of scenarios. In all cases, an overall "wall clock" time limit of one hour per instance was imposed.

5.6.1 *Benchmark Instances*

Existing benchmark instances for the TWAVRP focus solely on demand uncertainty. For the continuous setting, the authors of [242] introduced 40 randomly generated instances. The number of customers (n) in these instances varies from 10 to 25. Subsequently, [94] proposed 50 additional instances with n varying from 30 to 50. Each of the 90 available instances consists of 3 demand scenarios (low, medium, high), each with equal probability of occurrence. The average demand (for each customer) across the three scenarios is about $1/6$ of the vehicle capacity Q. The exogenous time windows are designed to be much wider than the endogenous time windows; in particular, the average (across customers)

exogenous time window width ($\ell_i - e_i$) is 10.8, compared to an endogenous time window width (w_i) of just 2.0. For the discrete setting, the authors of [241] introduced 80 randomly generated instances with n varying from 10 to 60. Except for the structure of the feasible time window set, the instances share similar characteristics as in the continuous setting, each consisting of 3 demand scenarios. In each instance, the number of candidate time windows (N_i) is equal to 3 for about 10% of the customers, 5 for about 60% of the customers, and 7 for the remaining 30%. Similarly to the continuous setting, the customer locations are generated using a uniform distribution over a square with the depot located in the center. Moreover, the time windows and vehicle capacities are chosen such that no more than eight customers can be visited on a single vehicle route in any scenario. All of the aforementioned test instances are inspired from the Dutch retail sector, and can be found online at http://people.few.eur.nl/spliet.

To test our algorithm on instances containing a large number of scenarios, we generated 80 additional benchmark instances. Specifically, for each existing (continuous and discrete) TWAVRP instance with $n \leq 25$, we used a similar procedure as described in [241] to generate 15 additional demand scenarios. For a particular instance, we first generate a nominal demand $\bar{q}_i$ for each customer $i \in V_C$ using a normal distribution with mean 5.0 and variance 1.5. To generate additional scenarios, we draw additive disturbances ϵ_{si} from a uniform distribution on $[-1.5, 1.5]$ for each $i \in V_C$ and $s \in \mathcal{S}$, and multiplicative factors f_s from a uniform distribution on $[0.625, 1.375]$ for each $s \in \mathcal{S}$. The demand of customer i in scenario s is then computed as $q_{si} = \left\lceil \max\left\{ f_s \left(\bar{q}_i + \epsilon_{si} \right), 10^{-6} \right\} \right\rceil$. The multiplicative factors f_s determine the level of correlation among the customer demands. For instance, high (low) values of f_s may represent the behavior that demands increase (decrease) uniformly for all retailers in a supply chain, whereas values close to one represent the nominal situation in which the demands are uncorrelated. The additional benchmark instances can be downloaded from `http://gounaris.cheme.cmu.edu/datasets/twavrp`.

5.6.2 *Comparison with Existing Methods*

We first compare the performance of our algorithm with the results published in [94] for the case of the continuous TWAVRP. We do not compare with the algorithm of [242], since the authors of [94] have demonstrated that their algorithm is superior to the former. Table 5.1 summarizes the comparison of the numerical performance across all 90 instances that are available for the continuous TWAVRP. The column **#** denotes the number of test instances that contain n customers. For each algorithm, **Optimal** denotes the number of test instances that it could solve to optimality in one hour while **Time (sec)** denotes the average time in seconds to solve these instances to optimality. For those instances which could not be solved to optimality in one hour, the column **Gap (%)** reports the average optimality gap, defined as $(UB - LB)/UB \times 100\%$, where LB and UB are respectively the global lower and upper bounds determined by the algorithm after one hour. The two methods are also compared in the performance

profiles [105] of Figure 5.8a. Our proposed algorithm is able to solve all but one (89 out of 90) benchmark instances to optimality, utilizing an average computation time of 169 seconds; of these, 32 instances were unsolved by the best previous method, while the one unsolved instance was determined to be within 0.8% of optimality. These results demonstrate that our algorithm strongly outperforms the existing method, solving more problems and achieving (or matching) the fastest computation time in all instances.

Table 5.1: Computational comparison of the proposed algorithm against the existing state-of-the-art algorithm (DS18) [94] on all 90 benchmark instances of the continuous TWAVRP.

		DS18			Proposed algorithm		
n	#	Optimal	Time (sec)	Gap (%)	Optimal	Time (sec)	Gap (%)
10	10	10	0.1	–	10	0.1	–
15	10	10	4.6	–	10	0.6	–
20	10	10	2.2	–	10	1.5	–
25	10	10	12.4	–	10	8.6	–
30	10	9	204.5	1.66	10	48.2	–
35	10	6	152.8	0.89	9	51.5	0.79
40	10	2	1860.0	1.16	10	342.3	–
45	10	0	–	2.74	10	361.3	–
50	10	0	–	4.26	10	696.0	–
All	90	57	117.0	2.56	89	169.1	0.79
Processor			Intel i7 3.5 GHz			Xeon E5-2687W 3.1GHz	

We now turn our attention to the discrete TWAVRP. Table 5.2 compares the numerical performance of our algorithm with the results published in [241] across all 80 instances of the discrete TWAVRP. The columns in this table have the same meaning as in Table 5.1. The two algorithms are also compared in the performance profiles of Figure 5.8b. Our algorithm is able to solve 54 out of 80 benchmark instance to optimality, utilizing an average computation time of 274 seconds; of these, 22 instances were unsolved by the best previous method, while the remaining unsolved instances were determined to be within 1.2% of optimality, on average. As in the continuous setting, our algorithm strongly outperforms the existing method: it solves more instances and achieves the fastest computation time in all of them.

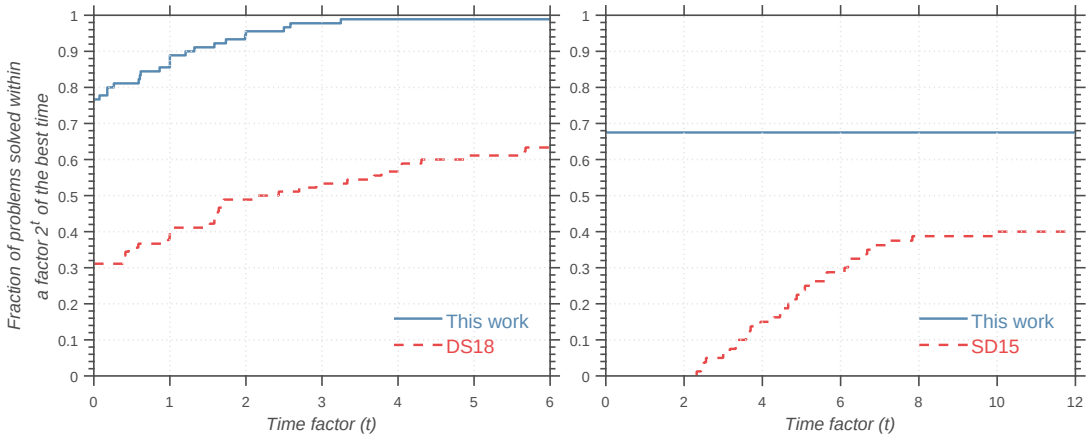
5.6.3 *Detailed Discussion of Results*

A comparison of Tables 5.1 and 5.2 shows that the discrete TWAVRP instances take longer to solve than the continuous ones. This can be partly explained by the fact that, in the continuous setting, the separation problem (5.7) is a linear program, while in the

Table 5.2: Computational comparison of the proposed algorithm against the existing state-of-the-art algorithm (SD15) [241] on all 80 benchmark instances of the discrete TWAVRP.

		SD15			Proposed algorithm		
n	#	Optimal	Time (sec)	Gap (%)	Optimal	Time (sec)	Gap (%)
10	10	10	3.9	–	10	0.1	–
15	10	10	185.9	–	10	15.2	–
20	10	9	1247.6	0.06	10	33.8	–
25	10	3	504.4	n/a†	9	248.8	0.43
30	10	0	–	n/a†	9	581.7	0.13
40	10	0	–	n/a†	5	1263.5	1.22
50	10	0	–	n/a†	1	533.6	1.31
60	10	0	–	n/a†	0	–	1.30
All	80	32	457.5	n/a†	54	274.4	1.21
Processor		Intel Core i5-2450M 2.5 GHz			Xeon E5-2687W 3.1GHz		

Note. The reported results for "SD15" are the best entries of Tables 3 and 4 from that publication [241].
† The optimality gaps for the unsolved instances have not been reported in the publication.



(a) Continuous (b) Discrete

Figure 5.8: Log-scaled performance profiles across all benchmark instances. The left graph compares the performance in the continuous setting, in which "DS18" shows the performance of the algorithm of [94], while the right graph compares the performance in the discrete setting, in which "SD15" shows the performance of the algorithm of [241]. In both graphs, "This work" shows the performance of our proposed algorithm. For each curve (i.e., algorithm), the value at $t = 0$ gives the fraction of benchmark instances for which it is fastest, while the limiting value at $t \to \infty$ gives the fraction of instances which it could solve within the time limit of one hour.

discrete setting, it is a mixed-integer linear program. Consequently, the algorithm spends a greater fraction of the total time in solving the separation problem in the latter case (see Table 5.3).

Table 5.3: Percentage of computing time spent in various parts of the algorithm (averaged across instances solved to optimality in one hour). "Solving VRPTW" and "Separation problem" refer to Steps 3′ and 5 of the algorithm from Section 5.4.2 respectively, while "Upper bounding" refers to the steps in Section 5.4.3.

	Continuous			Discrete		
n	Solving VRPTW	Separation problem	Upper bounding	Solving VRPTW	Separation problem	Upper bounding
25	84.6	0.0	15.0	81.0	15.4	3.3
30	86.8	0.1	12.9	75.7	21.4	2.7
40	89.2	0.2	10.4	90.6	8.2	1.2
50	93.1	0.0	6.8	95.7	1.6	2.8
≥ 25	89.0	0.1	10.8	81.6	15.6	2.6

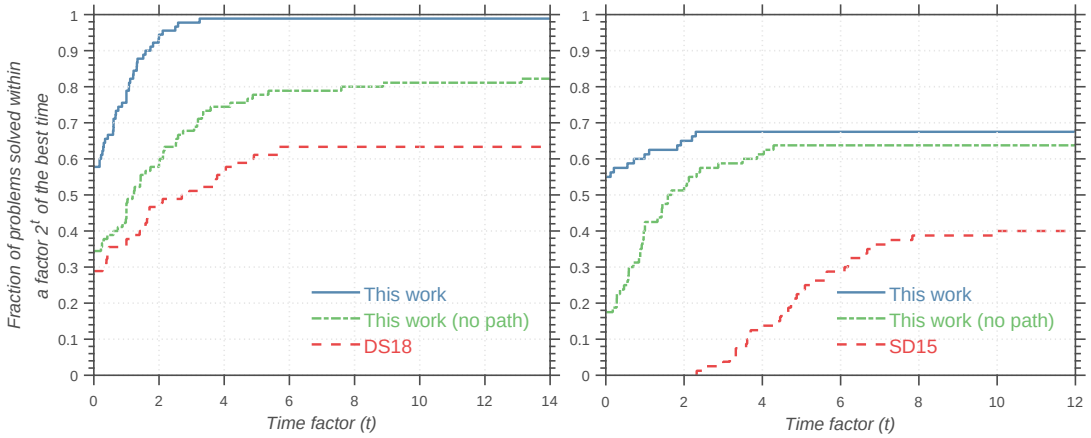
To show the efficacy of the path-based disjunctions and the associated branching rules (see Section 5.4.2), we disable them and run only the basic version of the algorithm from Section 5.4.1. Table 5.4 compares the performance of this basic version with the one incorporating the path-based disjunctions. They are also compared in the performance profiles of Figure 5.9. The results indicate that the path-based branching rules are important to improve the tractability of the overall algorithm. In particular, they are essential in reducing the total number of nodes that are explored in the overall search tree. We remark, however, that this reduction comes at a price: it requires modifying the underlying VRPTW solver (see Section 5.5.1). Nevertheless, even without the path-based branching rules, the basic version of our algorithm outperforms the existing ones (see Figure 5.9), while having the advantage of being able to utilize any VRPTW solver in a modular fashion.

Tables 5.5 and 5.6 present detailed results on all benchmark instances of the continuous and discrete TWAVRP, respectively. In these tables, if an instance could be solved to optimality within one hour, then **Opt [UB]** reports the corresponding optimal objective value, while **Time (sec) [LB]** reports the time to solve the instance to optimality. Otherwise, the columns respectively report in brackets the best upper and lower bounds found within the time limit of one hour.

Finally, it is worth noting that the average number of nodes processed and the average computation time elapsed until the optimal solution was found were about 91% and 93% of their respective totals, while both median percentages were 100%. These observations indicate that the algorithm most often terminated as soon as the optimal solution was

Table 5.4: Computational comparison of the algorithm with and without the path-based disjunctions on all 170 benchmark instances of the continuous and discrete TWAVRP.

		Without path-based disjunctions				With path-based disjunctions			
n	#	Optimal	Nodes	Time (sec)	Gap (%)	Optimal	Nodes	Time (sec)	Gap (%)
[10, 15]	40	40	5,887	91.5	–	40	26	4.0	–
[20, 25]	40	36	933	147.7	0.17	39	158	68.7	0.43
[30, 35]	30	22	949	327.6	0.16	28	132	220.7	0.46
[40, 45]	30	19	469	210.7	0.66	25	139	534.1	1.22
[50, 60]	30	8	557	1,061.0	1.12	11	105	681.2	1.30
All	170	125	2,427	229.4	0.75	143	108	208.9	1.19



(a) Continuous (b) Discrete

Figure 5.9: Log-scaled performance profiles across all benchmark instances. The profile "This work (no path)" refers to our proposed algorithm without the path-based disjunctions (see Section 5.4.2). The other profiles as well as the axes have the same meaning as in Figure 5.8.

detected and the upper bound updated, alluding to the fact that our algorithm generates strong lower bounds in Steps 3 and 3′.

5.6.4 *Instances containing a Large Number of Scenarios*

We now turn our attention to benchmark instances containing a large number of scenarios. Our goals are two-fold. First, we aim to understand how our algorithm performs as the number of considered scenarios (S) increases. Second, we aim to understand the cost benefits of considering more scenarios during strategic time window assignment.

Table 5.5: Detailed performance of the proposed algorithm on all 90 benchmark instances of the continuous TWAVRP.

Instance	n	Opt [UB]	Time (sec) [LB]	Instance	n	Opt [UB]	Time (sec) [LB]	Instance	n	Opt [UB]	Time (sec) [LB]
1	10	17.65	0.0	31	25	31.43	2.6	61	40	46.13*	9.7
2	10	15.56	0.2	32	25	30.71	3.0	62	40	48.35	27.8
3	10	17.42	0.0	33	25	33.71	9.9	63	40	44.48*	18.2
4	10	18.51	0.2	34	25	33.34	3.4	64	40	43.75	192.2
5	10	16.07	0.1	35	25	29.05	3.1	65	40	43.39*	20.2
6	10	18.00	0.0	36	25	30.50	3.9	66	40	44.68*	62.7
7	10	17.02	0.0	37	25	28.68	17.2	67	40	46.88*	110.2
8	10	23.89	0.1	38	25	35.69	38.5	68	40	44.96*	2,506.0
9	10	20.31	0.1	39	25	32.55	2.5	69	40	43.07*	31.1
10	10	16.31	0.1	40	25	32.14	2.0	70	40	43.00*	445.3
11	15	17.78	0.6	41	30	36.38	2.7	71	45	50.65*	20.4
12	15	27.10	1.3	42	30	34.69*	9.7	72	45	51.74*	69.4
13	15	29.37	0.4	43	30	35.48	285.4	73	45	41.70*	39.6
14	15	23.18	1.9	44	30	35.88	19.9	74	45	47.77*	228.6
15	15	24.15	0.3	45	30	35.55	26.1	75	45	49.39*	582.0
16	15	21.03	0.3	46	30	37.47	11.8	76	45	49.83*	1,082.5
17	15	22.04	0.3	47	30	32.54	5.7	77	45	51.09*	1,241.8
18	15	22.30	0.4	48	30	36.32	7.0	78	45	53.33*	102.2
19	15	26.52	0.4	49	30	35.30	67.6	79	45	48.09*	99.4
20	15	22.11	0.3	50	30	40.27	46.2	80	45	50.26*	146.8
21	20	28.08	0.8	51	35	43.46	102.9	81	50	58.11*	559.8
22	20	29.80	0.7	52	35	41.84	25.5	82	50	52.61*	211.0
23	20	30.30	1.0	53	35	45.03*	39.3	83	50	58.58*	2,826.9
24	20	24.16	1.3	54	35	41.54*	42.8	84	50	53.92*	115.7
25	20	29.84	7.8	55	35	37.92	12.7	85	50	54.96*	1,113.0
26	20	29.72	0.8	56	35	44.49*	17.9	86	50	52.83*	306.1
27	20	26.48	1.0	57	35	[41.04]	[40.72]	87	50	53.71*	93.3
28	20	26.14	0.8	58	35	41.22	64.1	88	50	56.12*	203.3
29	20	26.61	0.6	59	35	43.43	14.8	89	50	60.23*	1,299.0
30	20	26.36	0.6	60	35	42.27	143.1	90	50	58.93*	231.6

*Instances solved to optimality for the first time are indicated with an asterisk.

Table 5.6: Detailed performance of the proposed algorithm on all 80 benchmark instances of the discrete TWAVRP.

Instance	n	Opt [UB]	Time (sec) [LB]	Instance	n	Opt [UB]	Time (sec) [LB]	Instance	n	Opt [UB]	Time (sec) [LB]
1	10	12.83	0.1	31	25	35.47	24.3	61	50	[52.23]	[51.57]
2	10	16.84	0.2	32	25	32.66*	16.0	62	50	[55.61]	[55.19]
3	10	16.60	0.1	33	25	[31.75]	[31.61]	63	50	[50.75]	[50.09]
4	10	15.96	0.2	34	25	34.14*	236.9	64	50	51.17*	533.6
5	10	19.65	0.2	35	25	30.29*	617.7	65	50	[54.11]	[53.44]
6	10	18.13	0.3	36	25	32.54*	611.6	66	50	[57.52]	[56.57]
7	10	12.17	0.1	37	25	27.48*	568.5	67	50	[58.14]	[57.54]
8	10	17.09	0.2	38	25	34.83	15.9	68	50	[56.37]	[55.27]
9	10	20.14	0.1	39	25	34.39*	123.7	69	50	[53.85]	[53.40]
10	10	17.17	0.1	40	25	30.73	25.0	70	50	[57.37]	[56.37]
11	15	23.04	7.4	41	30	36.39*	37.9	71	60	[64.83]	[63.60]
12	15	25.27	1.0	42	30	40.59*	245.4	72	60	[62.60]	[61.57]
13	15	22.12	2.8	43	30	37.18*	88.7	73	60	[64.92]	[63.91]
14	15	18.46	0.7	44	30	[38.02]	[37.97]	74	60	[69.14]	[68.59]
15	15	24.87	129.9	45	30	36.72*	311.2	75	60	[63.61]	[63.15]
16	15	19.82	2.4	46	30	34.76*	237.8	76	60	[64.49]	[63.77]
17	15	21.96	4.7	47	30	42.24*	133.6	77	60	[61.24]	[60.82]
18	15	22.93	0.7	48	30	37.04*	3,501.3	78	60	[64.77]	[63.06]
19	15	23.14	1.3	49	30	40.47*	202.6	79	60	[65.48]	[64.91]
20	15	18.84	0.7	50	30	39.89*	477.2	80	60	[64.42]	[63.76]
21	20	27.99	1.2	51	40	[41.96]	[41.44]				
22	20	25.63	16.8	52	40	[47.43]	[47.37]				
23	20	26.53	199.2	53	40	41.76*	1,829.8				
24	20	32.36	3.9	54	40	45.96*	1,385.2				
25	20	28.84	4.7	55	40	[48.68]	[48.10]				
26	20	26.99	6.3	56	40	[44.88]	[43.99]				
27	20	27.55*	80.0	57	40	43.90*	619.3				
28	20	26.53	14.4	58	40	43.09*	918.1				
29	20	29.49	8.6	59	40	[49.27]	[48.50]				
30	20	23.55	2.5	60	40	47.13*	1,565.2				

*Instances solved to optimality for the first time are highlighted with an asterisk.

In pursuit of these goals, we consider the 80 benchmark instances consisting of 15 demand scenarios each (see Section 5.6.1). For each of these instances, we obtain time window assignments using our algorithm by considering the following sample average approximations: *(i)* the original instance with all 15 scenarios, and *(ii)* the original instance with only the first S scenarios, where $S \in \{1, 3, 5, 10\}$. For each approximation, we implement a parallel version of our algorithm in which Step 3′ and the upper bounding step in Section 5.4.3 are each parallelized using up to $\min\{S, 10\}$ threads.

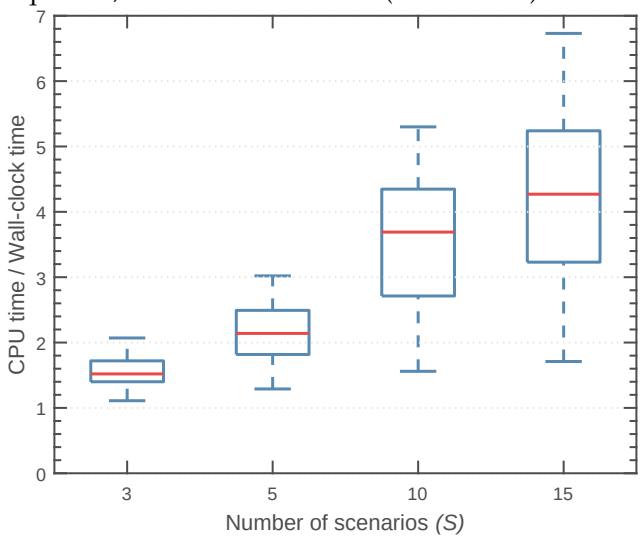
For $S = 1$ and $S = 3$, all 80 instances were solved to optimality, while 73 out of 80 instances were solved to optimality for $S = 5$ (the average gap for the 7 unsolved instances being less than 0.2%). Therefore, at the interest of brevity, we shall not show tabulated results for these cases. On the other hand, for the cases with the higher number of scenarios, namely $S = 10$ and $S = 15$, Table 5.7 presents a summary of the performance. The columns in this table have the same interpretation as in Tables 5.1 and 5.2; the only difference is that the column **Time (sec)** is now broken into two parts: **Wall** denotes the average wall clock time, while **CPU** denotes the average CPU time. The ratio between these quantities is a good measure of how well the algorithm scales across multiple threads (i.e., how much it benefits from parallelism), which is also plotted as a function of S in Figure 5.10.

Table 5.7: Summary of computational performance of the parallelized algorithm on 80 benchmark instances of the (continuous and discrete) TWAVRP, each containing S scenarios.

		$S = 10$				$S = 15$			
			Time (sec)				Time (sec)		
n	#	Optimal	Wall	CPU	Gap %	Optimal	Wall	CPU	Gap %
10	20	20	0.7	3.8	–	20	1.1	7.0	–
15	20	18	127.7	474.6	0.21	18	441.7	1,639.2	0.64
20	20	16	266.4	779.8	0.46	11	426.2	1,403.3	0.33
25	20	5	465.3	841.6	0.60	2	2,189.9	3,777.1	0.70
All	80	59	150.9	428.9	0.53	51	334.2	1,032.1	0.58

Table 5.7 shows that we can consistently solve all benchmark instances with 15 scenarios to an optimality gap of less than 1% within a time limit of one hour. Figure 5.10 shows that the speedup in wall clock time is sublinear with respect to the number of parallel threads. The reasons for deviating from a perfect linear speedup are two-fold. First, each node of our search tree does not necessarily require the solution of S VRPTW instances (see concluding paragraph of Section 5.4.1). Indeed, for the considered benchmark instances, the average number of VRPTW instances solved in a typical node is smaller than $S/2$. Second, the variance in solution times across the VRPTW instances solved in a node is typically large because the feasible route sets in a particular scenario might be drastically different compared to other scenarios in the same node (because of different demand

Figure 5.10: The ratio of CPU time to wall clock time with increasing number of scenarios (for instances whose solution took at least one wall clock second and at most one wall clock hour). For each S, the lower and upper most dashes denote the minimum and maximum values of the ratio, the lower and upper edges of the box denote the first and third quartile, while the middle bar (in red color) denotes the median ratio.



realizations). Consequently, the different CPU threads are not necessarily balanced, i.e., do not perform equal amount of work. Nevertheless, as Figure 5.10 shows, on average, our algorithm performs four times as many computations for instances containing 15 scenarios by utilizing up to ten threads as compared to using just one.

Finally, Table 5.8 shows the cost savings in routing that are to be expected by considering more than just one scenario during time window assignment. We calculate these savings as follows. First, for a particular instance (with a given S), we obtain the optimal time window assignment $\tau^\star$ from our algorithm. Next, we calculate the "in-sample" expected costs by *(i)* fixing the time windows to $\tau^\star$, *(ii)* solving a VRPTW instance with time windows fixed to $\tau^\star$ for each of the 15 postulated demand scenarios in the original benchmark instance, and *(iii)* averaging the costs. The "out-of-sample" expected costs are obtained in exactly the same manner except that in step *(ii)*, the costs are evaluated over 100 independently generated demand scenarios, which are randomly drawn using the procedure described in Section 5.6.1. Table 5.8 shows that, on average, we expect to do better than the deterministic solution by about 2.3% when considering up to 3 scenarios and by about 3.2% when considering up to 15 scenarios (based on out-of-sample evaluations). If we consider only the discrete instances, the out-of-sample savings increase to about 3.0% for $S = 3$ and 3.7% for $S = 15$. In either case, we observe that the marginal benefits diminish as the number of scenarios grows. Although we expect this trend to generally hold, we also expect the actual magnitudes of the cost savings to depend on the problem parameters as well as the dimensionality of the uncertainty (the number of customers in this case). Indeed, for high-dimensional problems with

narrow and discrete time windows, we expect greater cost benefits by considering more scenarios of the uncertainty.

Table 5.8: Expected cost savings from considering S scenarios during strategic time window assignment relative to considering only one scenario (averaged across all 60 instances with $n \geq 15$).

	$S = 1$	$S = 3$	$S = 5$	$S = 10$	$S = 15$
In-sample	0.00%	2.26%	2.88%	3.13%	3.22%
Out-of-sample	0.00%	2.36%	2.89%	3.18%	3.21%

5.7 SUMMARY

This chapter addresses the challenge of dealing with operational uncertainty during strategic decision-making in the context of vehicle routing. In particular, we studied problems in which the decisions correspond to an allocation of long-term delivery time windows to customers. These problems are motivated from several real-world distribution operations, and are particularly common in retail. We proposed a novel algorithm to solve this problem that is highly competitive with existing methods. It draws on existing algorithms for deterministic vehicle routing problems (both exact and heuristic) and can use any vehicle routing solver as a subroutine, thus facilitating its deployment in practice. From a modeling viewpoint, it allows the user to postulate potential scenarios of future uncertainty corresponding to different routing-specific parameters, as well as incorporate modular changes in routing-specific constraints. Our business insights, aided via numerical experiments, are that long-term costs are expected to decrease by modeling more scenarios of future uncertainty but the marginal benefits rapidly diminish as a function of the number of postulated scenarios. In other words, a few scenarios are sufficient to obtain time window assignments that would incur lower long-term costs compared to assignments that would be obtained by completely ignoring operational uncertainty.

5.8 APPENDIX: NOMENCLATURE

n	Number of customers
$G = (V, A)$	Directed graph with node set V and arc set A
Q	Capacity of each vehicle
c_{ij}	Routing cost along arc $(i, j) \in A$
t_{ij}	Travel time along arc $(i, j) \in A$
$[e_i, \ell_i]$	Exogenous time window of node $i \in V$
V_C	Set of customers
q_i	Demand of customer $i \in V_C$
u_i	Service time of customer $i \in V_C$
TW_i	Feasible time window set of customer $i \in V_C$
V_{cont}	Subset of customers with continuous time window sets
V_{disc}	Subset of customers with discrete time window sets
w_i	Width of endogenous time window of customer $i \in V_{\text{cont}}$
N_i	Number of candidate time windows of customer $i \in V_{\text{disc}}$
$[\underline{y}_{ib}, \overline{y}_{ib}]$	b^{th} candidate time window of customer $i \in V_{\text{disc}}$
S	Number of scenarios
$\mathcal{S}$	Set of scenarios
p_s	Probability of scenario $s \in \mathcal{S}$
θ_s	Realization of the operational parameters in scenario $s \in \mathcal{S}$
$\mathbf{R}$	Route set that partitions the customer set V_C
$c(\mathbf{R})$	Cost of route set $\mathbf{R}$
$\mathcal{X}(\mathbf{R}, \tau; \theta)$	Set of feasible arrival time vectors corresponding to route set $\mathbf{R}$, time windows τ_i for each $i \in V_C$ and operational parameters θ
$\mathcal{R}(\tau; \theta)$	Set of all feasible route sets corresponding to time windows τ_i for each $i \in V_C$ and operational parameters θ
Ω	Set of all elementary vehicle routes that are feasible with respect to capacity and time window constraints
$\mathcal{F}$	Collection of forbidden paths
$\texttt{VRPTW}(\tau; \theta)$	Optimal value of the vehicle routing problem with time windows τ_i for each $i \in V_C$ and operational parameters θ
$\texttt{VRPTWFP}(\tau, \mathcal{F}; \theta)$	Optimal value of the vehicle routing problem with time windows τ_i for each $i \in V_C$, forbidden paths $\mathcal{F}$ and parameters θ
$\mathbb{1}_{[\mathcal{E}]}$	Indicator function taking a value of 1 if the expression $\mathcal{E}$ is true and 0 otherwise

6

K-ADAPTABILITY IN TWO-STAGE ROBUST OPTIMIZATION

Chapters 2 and 3 applied robust optimization to solve vehicle routing problems under uncertainty. In particular, Chapter 2 applied the theory of static robust optimization, while Chapter 3 applied the theory of two-stage robust optimization. It turns out that the underlying optimization problems in both chapters are special cases of a very general class of dynamic decision-making problems, in which decisions need to be made both in anticipation of and in response to the realization of unknown problem parameters.

In this chapter, we study generic two-stage robust optimization problems with mixed discrete-continuous decisions and mixed discrete-continuous uncertain parameters. These problems have a broad range of applications, including but not limited to just vehicle routing operations. However, their potential applicability is currently restricted because of two fundamental challenges that they pose: *(i)* they constitute infinite-dimensional problems that require a finite-dimensional approximation, and *(ii)* the presence of discrete recourse decisions typically prohibits duality-based solution schemes.

The aim of this chapter is to address these challenges from a theoretical, algorithmic and practical perspective. We address the first challenge by studying a K-adaptability formulation that selects K candidate recourse policies *before* observing the realization of the uncertain parameters and that implements the best of these policies *after* the realization is known. We address the second challenge by developing an algorithmic scheme that enjoys strong convergence properties, both in theory and in practice.

This chapter is organized as follows. First, the K-adaptability problem is motivated and formally defined in Section 6.1. Section 6.2 then analyzes its geometry and tractability, and it shows how the K-adaptability problem can be used to model continuous recourse decisions via a novel, highly flexible class of decision rules that generalize classical affine decision rules. Section 6.3 develops a branch-and-bound algorithm for the K-adaptability problem and analyzes its convergence. Numerical results are then presented in Section 6.4, and we close with a summary of the main results from this chapter in Section 6.5.

6.1 MOTIVATION AND BACKGROUND

Dynamic decision-making under uncertainty, where actions need to be taken both in anticipation of and in response to the realization of a priori uncertain problem parameters, arguably forms one of the most challenging domains of operations research and optimization theory. Despite intensive research efforts over the past six decades, many uncertainty-affected optimization problems resist solution, and even our understanding of the complexity of these problems remains incomplete.

In the last two decades, robust optimization has emerged as a promising methodology to counter some of the intricacies associated with decision-making under uncertainty. The rich theory on static robust optimization problems, in which all decisions have to be taken before the uncertainty is resolved, is summarized in [36, 42, 121]. However, dynamic robust optimization problems, in which some of the decisions can adapt to the observed uncertainties, are still poorly understood.

This chapter is concerned with *two-stage robust optimization problems* of the form

$$\inf_{\boldsymbol{x}\in\mathcal{X}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \tag{6.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^{N_1}$, $\mathcal{Y} \subseteq \mathbb{R}^{N_2}$ and $\Xi \subseteq \mathbb{R}^{N_p}$ constitute nonempty and bounded mixed-integer linear programming (MILP) representable sets, $\boldsymbol{c} \in \mathbb{R}^{N_1}$, and the functions $\boldsymbol{d} : \Xi \mapsto \mathbb{R}^{N_2}$, $\boldsymbol{T} : \Xi \mapsto \mathbb{R}^{L\times N_1}$, $\boldsymbol{W} : \Xi \mapsto \mathbb{R}^{L\times N_2}$ and $\boldsymbol{h} : \Xi \mapsto \mathbb{R}^{L}$ are affine. In problem (6.1), the vector $\boldsymbol{x}$ represents the first-stage (or 'here-and-now') decisions which are taken before the value of the uncertain parameter vector $\boldsymbol{\xi}$ from within the uncertainty set Ξ is observed. The vector $\boldsymbol{y}$, on the other hand, denotes the second-stage (or 'wait-and-see') decisions that can adapt to the realized value of $\boldsymbol{\xi}$. We emphasize that problem (6.1) can have a random recourse, i.e., the recourse matrix $\boldsymbol{W}$ may depend on the uncertain parameters $\boldsymbol{\xi}$. Moreover, we do not assume a relatively complete recourse; that is, for some first-stage decisions $\boldsymbol{x} \in \mathcal{X}$, there can be parameter realizations $\boldsymbol{\xi} \in \Xi$ such that there is no feasible second-stage decision $\boldsymbol{y}$. Also, we do not assume that the sets $\mathcal{X}$, $\mathcal{Y}$ or Ξ are convex.

Remark 6.1 (Uncertain First-Stage Objective Coefficients)**.** *The assumption that $\boldsymbol{c}$ is deterministic does not restrict generality. Indeed, problem (6.1) accounts for uncertain first-stage objective coefficients $\boldsymbol{c}' : \Xi \mapsto \mathbb{R}^{N_1}$ if we augment the second-stage decisions $\boldsymbol{y}$ to $(\boldsymbol{y}, \boldsymbol{y}')$, replace the second-stage objective coefficients $\boldsymbol{d}$ with $(\boldsymbol{d}, \boldsymbol{c}')$ and impose the constraint that $\boldsymbol{y}' = \boldsymbol{x}$.*

Even in the special case where $\mathcal{X}$, $\mathcal{Y}$ and Ξ are linear programming (LP) representable, problem (6.1) involves infinitely many decision variables and constraints, and it has been shown to be NP-hard [143]. Nevertheless, problem (6.1) simplifies considerably if the sets $\mathcal{Y}$ and Ξ are LP representable. For this setting, several approximate solution schemes have been proposed that replace the second-stage decisions with *decision rules*, i.e., parametric classes of linear or nonlinear functions of $\boldsymbol{\xi}$ [38, 79, 128, 131, 174]. If we further assume that $\boldsymbol{d}$, $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic and Ξ is of simple form (e.g., a budget

uncertainty set), a number of exact solution schemes based on Benders' decomposition [49, 163, 251, 279] and semi-infinite programming [20, 275] have been developed.

Problem (6.1) becomes significantly more challenging if the set $\mathcal{Y}$ is not LP representable. For this setting, conservative MILP approximations have been developed in [135, 263] by partitioning the uncertainty set Ξ into hyperrectangles and restricting the continuous and integer recourse decisions to affine and constant functions of $\boldsymbol{\xi}$ over each hyperrectangle, respectively. These a priori partitioning schemes have been extended to iterative partitioning approaches in [44, 218]. Iterative solution approaches based on decision rules have been proposed in [46, 47]. However, to the best of our knowledge, none of these approaches has been shown to converge to an optimal solution of problem (6.1). For the special case where problem (6.1) has a relatively complete recourse, $\boldsymbol{d}$, $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic and the optimal value of the second-stage problem is quasi-convex over Ξ, the solution scheme of [275] has been extended in [280] to a nested semi-infinite approach that can solve instances of problem (6.1) with MILP representable sets $\mathcal{Y}$ and Ξ to optimality in finite time.

Instead of solving problem (6.1) directly, we study its *K-adaptability problem*

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \left\{ \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top\boldsymbol{y}_k \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \tag{6.2}$$

where $\mathcal{Y}^K = \bigtimes_{k=1}^K \mathcal{Y}$ and $\mathcal{K} = \{1,\ldots,K\}$. Problem (6.2) determines K non-adjustable second-stage policies $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_K$ here-and-now and subsequently selects the best of these policies in response to the observed value of $\boldsymbol{\xi}$. If all policies are infeasible for some realization $\boldsymbol{\xi}\in\Xi$, then the solution $(\boldsymbol{x},\boldsymbol{y})$ attains the objective value $+\infty$. By construction, the K-adaptability problem (6.2) bounds the two-stage robust optimization problem (6.1) from above.

Our interest in problem (6.2) is motivated by two observations. Firstly, problem (6.2) has been shown to be a remarkably good approximation of problem (6.1), both in theory and in numerical experiments [43, 145]. Secondly, and perhaps more importantly, the K-adaptability problem conforms well with human decision-making, which tends to address uncertainty by developing a small number of contingency plans, rather than devising the optimal response for every possible future state of the world. For instance, practitioners may prefer a limited number of contingency plans to full flexibility in the second stage for operational (e.g., in production planning or logistics) or organizational (e.g., in emergency response planning) reasons.

The K-adaptability problem was first studied in [43], where the authors reformulate the 2-adaptability problem as a finite-dimensional bilinear program and solve it heuristically. The authors also show that the 2-adaptability problem is NP-hard even if $\boldsymbol{d}$, $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic, and they develop necessary conditions for the K-adaptability problem (6.2) to outperform the static robust problem (where all decisions are taken here-and-now). The relationship between the K-adaptability problem (6.2) and static robust optimization is further explored in [48] for the special case where $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic. The

authors show that the gaps between both problems and the two-stage robust optimization problem (6.1) are intimately related to geometric properties of the uncertainty set Ξ. Finite-dimensional MILP reformulations for problem (6.2) are developed in [145] under the additional assumption that both the here-and-now decisions $\boldsymbol{x}$ and the wait-and-see decisions $\boldsymbol{y}$ are binary. The authors show that both the size of the reformulations as well as their gaps to the two-stage robust optimization problem (6.1) depend on whether the uncertainty only affects the objective coefficients $\boldsymbol{d}$, or whether the constraint coefficients $\boldsymbol{T}$, $\boldsymbol{W}$ and $\boldsymbol{h}$ are uncertain as well. Finally, it is shown in [68, 69] that for polynomial time solvable deterministic combinatorial optimization problems, the associated instances of problem (6.2) without first-stage decisions $\boldsymbol{x}$ can also be solved in polynomial time if all of the following conditions hold: *(i)* Ξ is convex, *(ii)* only the objective coefficients $\boldsymbol{d}$ are uncertain, and *(iii)* $K > N_2$ policies are sought. This result has been extended to discrete uncertainty sets in [70], in which case pseudo-polynomial solution algorithms can be developed.

In this chapter, we expand the literature on the K-adaptability problem in two ways. From an *analytical viewpoint*, we compare the two-stage robust optimization problem (6.1) with the K-adaptability problem (6.2) in terms of their continuity, convexity and tractability. We also investigate when the approximation offered by the K-adaptability problem is tight, and under which conditions the two-stage robust optimization problem and the K-adaptability problem reduce to single-stage problems. From an *algorithmic viewpoint*, we develop a branch-and-bound scheme for the K-adaptability problem (6.2) that combines ideas from semi-infinite and disjunctive programming. We establish conditions for its asymptotic and finite time convergence; we show how it can be refined and integrated into state-of-the-art MILP solvers; and, we present a heuristic variant that can address large-scale instances. In contrast to existing approaches, our algorithm can handle mixed continuous and discrete decisions in both stages as well as discrete uncertainty, and allows for modeling continuous second-stage decisions via a novel class of highly flexible piecewise affine decision rules. Extensive numerical experiments on benchmark data from various application domains indicate that our algorithm is highly competitive with state-of-the-art solution schemes for problems (6.1) and (6.2).

Notation. Vectors and matrices are printed in bold lowercase and uppercase letters, respectively, while scalars are printed in regular font. We use $\mathbf{e}_k$ to denote the k^{th} unit basis vector and $\mathbf{e}$ to denote the vector whose components are all ones, respectively; their dimensions will be clear from the context. The i^{th} row vector of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{a}_i^\top$. For a logical expression $\mathcal{E}$, we define $\mathbb{I}[\mathcal{E}]$ as the indicator function which takes a value of 1 is $\mathcal{E}$ is true and 0 otherwise.

6.2 PROBLEM ANALYSIS

In this section, we analyze the geometry and tractability of the two-stage robust optimization problem (6.1) and its associated K-adaptability problem (6.2). To this end,

		Continuity	Convexity	Tractability
First-stage problem	problem (6.1)	if feasible	if $\mathcal{X}$, $\mathcal{Y}$ convex	if $\mathcal{X}$, $\mathcal{Y}$, Ξ convex and OBJ
	problem (6.2)	if feasible	typically not	if $\mathcal{X}$, $\mathcal{Y}$, Ξ convex and OBJ
Evaluation problem	problem (6.1)	if OBJ	if Ξ convex and OBJ	if Ξ, $\mathcal{Y}$ convex and OBJ
	problem (6.2)	if OBJ or CON	if Ξ convex and OBJ	if Ξ convex and OBJ
Second-stage problem	problem (6.1)	if feasible	if $\mathcal{Y}$ convex	if $\mathcal{Y}$ convex
	problem (6.2)	if feasible	always	always
	Propositions	6.1, 6.5	6.2, 6.6	6.3, 6.7

	Reduction to static problem	Optimality of problem (6.2)
problem (6.1)	if Ξ convex and OBJ	if $\mathcal{Y}$, Ξ convex and OBJ
problem (6.2)	if Ξ convex, OBJ and $K > \min\{N_2, N_p\}$	if Ξ convex, OBJ and $K > \min\{N_2, N_p\}$
	Propositions 6.4, 6.8	if $\lvert\mathcal{Y}\rvert$ finite and $K \geq \lvert\mathcal{Y}\rvert$
		Proposition 6.9

Table 6.1: Summary of theoretical results from Sections 6.2.1 and 6.2.2. Here, "OBJ" refers to instances where only the objective coefficients $\boldsymbol{d}$ are uncertain, while $\boldsymbol{T}$, $\boldsymbol{W}$ and $\boldsymbol{h}$ are constant. Similarly, "CON" refers to instances where only the constraint coefficients $\boldsymbol{T}$, $\boldsymbol{W}$ and right-hand sides $\boldsymbol{h}$ are uncertain, while $\boldsymbol{d}$ is constant.

Sections 6.2.1 and 6.2.2 characterize the continuity, convexity and tractability of both problems, as well as their relationship to the static robust optimization problem where all decisions are taken here-and-now. Section 6.2.3 shows how the K-adaptability problem with continuous second-stage decisions enables us to approximate the two-stage robust optimization problem (6.1) through highly flexible piecewise-affine decision rules.

Table 6.1 summarizes our theoretical results from Sections 6.2.1 and 6.2.2. In the table, the *first-stage problem* refers to the overall problems (6.1) and (6.2), the *evaluation problem* refers to the maximization over $\boldsymbol{\xi} \in \Xi$ for a fixed first-stage decision, and the *second-stage problem* refers to the inner minimization over $\boldsymbol{y} \in \mathcal{Y}$ or $k \in \mathcal{K}$ for a fixed first-stage decision and a fixed realization of the uncertain problem parameters. The table reveals that despite significant differences in their formulations, the problems (6.1) and (6.2) behave very similarly. The most significant difference is caused by the replacement of the optimization over the second-stage decisions $\boldsymbol{y} \in \mathcal{Y}$ in problem (6.1) with the selection of a candidate policy $k \in \mathcal{K}$ in problem (6.2). This ensures that the second-stage problem in (6.2) is always continuous, convex and tractable, whereas the first-stage problem in (6.2) fails to be convex even if $\mathcal{X}$ and $\mathcal{Y}$ are convex. Moreover, in contrast to problem (6.1), the evaluation problem in (6.2) remains continuous as long as either the objective function or the constraints are unaffected by uncertainty. For general problem instances, however, neither of the two evaluation problems is continuous. As we will

see in Section 6.3.2, this directly impacts the convergence of our branch-and-bound algorithm, which only takes place asymptotically in general. Note that the convexity of the problems (6.1) and (6.2) does not depend on the shape of the uncertainty set Ξ.

6.2.1 *Analysis of the Two-Stage Robust Optimization Problem*

To analyze problem (6.1), we equivalently rewrite it as

$$\inf_{\boldsymbol{x}\in\mathcal{X}} \ \boldsymbol{c}^\top\boldsymbol{x} + \mathcal{Q}(\boldsymbol{x}) \quad \text{with} \qquad \mathcal{Q}(\boldsymbol{x}) = \sup_{\boldsymbol{\xi}\in\Xi} \ Q(\boldsymbol{x},\boldsymbol{\xi}),$$
$$\text{where} \quad Q(\boldsymbol{x},\boldsymbol{\xi}) = \inf_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top\boldsymbol{y} \ : \ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\} \tag{6.1$'$}$$

for $\mathcal{Q} : \mathcal{X} \mapsto \mathbb{R}\cup\{+\infty\}$ and $Q : \mathcal{X}\times\Xi \mapsto \mathbb{R}\cup\{+\infty\}$.

We first investigate whether the infima and the supremum in problem (6.1′) are attained.

Proposition 6.1 (Continuity)**.** *Problem* (6.1′) *satisfies the following properties.*

(i) The problem $Q(\boldsymbol{x},\boldsymbol{\xi})$ attains its infimum, if it is feasible.

(ii) The problem $\mathcal{Q}(\boldsymbol{x})$ attains its supremum, if only the objective function is uncertain. Otherwise, $\mathcal{Q}(\boldsymbol{x})$ does not necessarily attain its supremum, even if only the constraint right-hand sides are uncertain.

(iii) The problem (6.1′) *attains its infimum, if it is feasible.*

Proof. The first statement holds since the problem $Q(\boldsymbol{x},\boldsymbol{\xi})$ minimizes an affine function in $\boldsymbol{y}$ over the intersection of the compact set $\mathcal{Y}$ and the polyhedron $\{\boldsymbol{y}\in\mathbb{R}^{N_2} \ : \ \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi}) - \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x}\}$.

In view of the second statement, assume first that only the objective function in problem (6.1′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi}\in\Xi$. Then the inner problem $Q(\boldsymbol{x},\boldsymbol{\xi})$ simplifies to $Q(\boldsymbol{x},\boldsymbol{\xi}) = \inf\{\boldsymbol{d}(\boldsymbol{\xi})^\top\boldsymbol{y} \ : \ \boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{W}}\}$ with $\mathcal{Y}_{\boldsymbol{W}} = \{\boldsymbol{y}\in\mathcal{Y} \ : \ \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} - \boldsymbol{T}\boldsymbol{x}\}$. If $\mathcal{Y}_{\boldsymbol{W}} = \emptyset$, then $Q(\boldsymbol{x},\boldsymbol{\xi}) = +\infty$ for all $\boldsymbol{\xi}\in\Xi$, and any $\boldsymbol{\xi}\in\Xi$ attains the supremum. Otherwise, we have $Q(\boldsymbol{x},\boldsymbol{\xi}) = \inf\{\boldsymbol{d}(\boldsymbol{\xi})^\top\boldsymbol{y} \ : \ \boldsymbol{y}\in \operatorname{ext\,conv}\mathcal{Y}_{\boldsymbol{W}}\}$, where $\operatorname{ext\,conv}\mathcal{Y}_{\boldsymbol{W}}$ denotes the set of (finitely many) extreme points of the convex hull of $\mathcal{Y}_{\boldsymbol{W}}$. We thus conclude from [223, Proposition 1.26] that $Q(\boldsymbol{x},\boldsymbol{\xi})$ is upper semicontinuous in $\boldsymbol{\xi}$ for every fixed $\boldsymbol{x}$, and [223, Theorem 1.9] then implies that $\mathcal{Q}(\boldsymbol{x})$ attains its supremum since Ξ is a compact set.

Assume now that only the constraint right-hand sides in problem (6.1′) are allowed to be uncertain, and consider the following instance of problem $\mathcal{Q}(\boldsymbol{x})$:

$$\sup_{\xi\in[0,1]} \ \inf_{\substack{\tau\in\mathbb{R},\\ y\in\{0,1\}}} \ \{\tau - y \ : \ y \leq \xi \leq \tau\}$$

The inner minimization problem is optimized by $\tau^\star = \xi$ and $y^\star = \lfloor \xi \rfloor$. The supremum is 1, and it is approached by the sequence of feasible solutions $\xi \uparrow 1$. Note, however, that the limit point $\xi^\star = 1$ of this sequence results in an objective function value of 0.

As for the third statement, we first note that the extended real-valued function

$$Q(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}) = \begin{cases} \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} & \text{if } \boldsymbol{y} \in \mathcal{Y} \text{ and } \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi}), \\ +\infty & \text{otherwise} \end{cases}$$

is lower semicontinuous in $(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y})$ since the set $\{(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}) \in \mathbb{R}^{N_1} \times \mathbb{R}^{N_p} \times \mathcal{Y} : \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi})\}$ is closed. From [223, Theorem 1.17] and [223, Proposition 1.26] we conclude that the lower semicontinuity is preserved by the partial minimization over $\boldsymbol{y}$ and the partial maximization over $\boldsymbol{\xi}$. By [223, Theorem 1.9] and the fact that $\mathcal{X}$ is compact we can then conclude that the problem (6.1′) attains its infimum whenever it is feasible. □

It is shown in [145, Example 1] that $\mathcal{Q}(\boldsymbol{x})$ may not attain its supremum if both the objective function and the constraint right-hand sides in problem (6.1′) are uncertain. Proposition 6.1 (ii) strengthens this observation to instances of problem (6.1′) where only the constraint right-hand sides are uncertain. We now consider the convexity properties of problem (6.1′).

Proposition 6.2 (Convexity)**.** *Problem (6.1′) satisfies the following properties.*

(i) The problem $Q(\boldsymbol{x}, \boldsymbol{\xi})$ is convex, if $\mathcal{Y}$ is convex.

(ii) The problem $\mathcal{Q}(\boldsymbol{x})$ is convex, if Ξ is convex and only the objective function is uncertain, irrespective of $\mathcal{Y}$. Otherwise, $\mathcal{Q}(\boldsymbol{x})$ is typically not convex, even if Ξ and $\mathcal{Y}$ are convex and only the constraint right-hand sides are uncertain.

(iii) The problem (6.1′) is convex, if $\mathcal{X}$ and $\mathcal{Y}$ are convex, irrespective of Ξ.

Proof. The first statement directly follows from the linearity of the objective function and the convexity of the feasible set.

In view of the second statement, assume first that Ξ is convex and only the objective function in problem (6.1′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi} \in \Xi$. Then the inner problem $Q(\boldsymbol{x}, \boldsymbol{\xi})$ simplifies to $Q(\boldsymbol{x}, \boldsymbol{\xi}) = \inf\{\boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} : \boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{W}}\}$ with $\mathcal{Y}_{\boldsymbol{W}} = \{\boldsymbol{y} \in \mathcal{Y} : \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} - \boldsymbol{T}\boldsymbol{x}\}$. Assume that $\mathcal{Y}_{\boldsymbol{W}} \neq \emptyset$; the other case is trivial. Then we have $Q(\boldsymbol{x}, \boldsymbol{\xi}) = \inf\{\boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} : \boldsymbol{y} \in \operatorname{ext\,conv} \mathcal{Y}_{\boldsymbol{W}}\}$, where ext conv $\mathcal{Y}_{\boldsymbol{W}}$ denotes the set of (finitely many) extreme points of the convex hull of $\mathcal{Y}_{\boldsymbol{W}}$. We thus conclude from [223, Proposition 2.9] that $Q(\boldsymbol{x}, \boldsymbol{\xi})$ is concave in $\boldsymbol{\xi}$ for every fixed $\boldsymbol{x}$, which implies that $\mathcal{Q}(\boldsymbol{x})$ is a convex optimization problem.

Assume now that the constraint right-hand sides in problem (6.1′) are allowed to be uncertain, and consider the following instance of problem $\mathcal{Q}(\boldsymbol{x})$:

$$\sup_{\xi \in [-1,1]} \inf_{y \in \mathbb{R}} \{y : y \geq \xi, \ y \geq -\xi\}$$

Since the inner minimization problem is optimized by $y^\star = |\xi|$, the problem maximizes the (convex) 1-norm of ξ over the interval $[-1, 1]$, which amounts to a non-convex optimization problem.

As for the third statement, assume that $\mathcal{X}$ and $\mathcal{Y}$ are convex, and consider the function

$$Q(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}) = \begin{cases} \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} & \text{if } \boldsymbol{y} \in \mathcal{Y} \text{ and } \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi}), \\ +\infty & \text{otherwise.} \end{cases}$$

The function $Q(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y})$ is convex in $(\boldsymbol{x}, \boldsymbol{y})$ for every fixed $\boldsymbol{\xi} \in \Xi$. From [223, Proposition 2.22] and [223, Proposition 2.9] we conclude that the convexity is preserved by the partial minimization over $\boldsymbol{y}$ and the partial maximization over $\boldsymbol{\xi}$. The problem (6.1′) thus minimizes the sum of two convex functions $\boldsymbol{c}^\top \boldsymbol{x}$ and $\mathcal{Q}(\boldsymbol{x})$ over the convex set $\mathcal{X}$, which is a convex optimization problem. □

One readily verifies that the third statement in Proposition 6.2 does not hold if $\mathcal{Y}$ is not convex. We now investigate under which conditions we can solve problem (6.1′) in polynomial time.

Proposition 6.3 (Tractability)**.** *Problem* (6.1′) *satisfies the following properties.*

(i) The problem $Q(\boldsymbol{x}, \boldsymbol{\xi})$ can be solved in polynomial time, if $\mathcal{Y}$ is convex.

(ii) The problem $\mathcal{Q}(\boldsymbol{x})$ can be solved in polynomial time, if Ξ and $\mathcal{Y}$ are convex and only the objective function is uncertain. Otherwise, $\mathcal{Q}(\boldsymbol{x})$ is strongly NP-hard, even if Ξ and $\mathcal{Y}$ are convex and only the constraint right-hand sides are uncertain.

(iii) The problem (6.1′) *can be solved in polynomial time, if $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and only the objective function is uncertain. Otherwise, the problem is strongly NP-hard, even if $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and only the constraint right-hand sides are uncertain.*

Proof. If $\mathcal{Y}$ is convex, then the problem $Q(\boldsymbol{x}, \boldsymbol{\xi})$ amounts to a linear program that can be solved in polynomial time. This shows the first statement.

The first part of the second statement follows from the proof of the first part of the third statement below if we fix $\mathcal{X} = \{\boldsymbol{x}\}$ and $\boldsymbol{c} = \boldsymbol{0}$ in problem (6.1′). For the second part of the second statement, we recall the strongly NP-hard 0/1 Integer Programming (IP) feasibility problem [122]:

0/1 INTEGER PROGRAMMING FEASIBILITY.

Instance. Given are $\boldsymbol{A} \in \mathbb{Z}^{R \times N_p}$ and $\boldsymbol{b} \in \mathbb{Z}^R$.
Question. Is there a vector $\boldsymbol{\xi} \in \{0, 1\}^{N_p}$ such that $\boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b}$?

We show that the IP feasibility problem has an affirmative answer if and only if the problem

$$\sup \left\{ \inf \left\{ \mathbf{e}^\top \boldsymbol{y} \,:\, \boldsymbol{y} \in \mathbb{R}^{N_p},\ \ \boldsymbol{y} \geq \boldsymbol{\xi} - \frac{1}{2}\mathbf{e},\ \ \boldsymbol{y} \geq \frac{1}{2}\mathbf{e} - \boldsymbol{\xi} \right\} \,:\, \boldsymbol{\xi} \in [0, 1]^{N_p},\ \ \boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b} \right\} \quad (6.3)$$

has an optimal value of $N_p/2$. Note that (6.3) can be interpreted as an instance of $\mathcal{Q}(\boldsymbol{x})$. For any fixed $\boldsymbol{\xi}$, the infimum in (6.3) evaluates to $\|\boldsymbol{\xi} - \mathbf{e}/2\|_1$. Thus, the optimal value of (6.3) is equal to $N_p/2$ if and only if there is $\boldsymbol{\xi} \in [0,1]^{N_p}$ satisfying $\|\boldsymbol{\xi} - \mathbf{e}/2\|_1 = N_p/2$ and $\boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b}$. The statement now follows from the fact that $\|\boldsymbol{\xi} - \mathbf{e}/2\|_1 = N_p/2$ if and only if $\boldsymbol{\xi} \in \{0,1\}^{N_p}$.

In view of the first part of the third statement, the proof of Proposition 6.4 below shows that, under the stated assumptions, problem (6.1′) reduces to a single-stage robust optimization problem. Dualizing the inner maximization problem in that single-stage reformulation allows us to solve the overall problem in polynomial time as a linear program, see [36]. Finally, the second part of the third statement follows from the proof of the second part of the second statement if we amend problem (6.3) by appending an outer infimum over the singleton set $\mathcal{X} = \{1\}$. □

The NP-hardness of the two-stage robust optimization problem (6.1′) has previously been established for the case where $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and only the constraint right-hand sides are uncertain [143, Theorem 3.5]. We provide an alternative proof in Proposition 6.3 (iii) to facilitate a self-contained comparison with the K-adaptability problem (6.2) in Proposition 6.7 (ii) below. Moreover, it is shown in [143, Theorem 3.3] that problem (6.1′) can be solved in polynomial time whenever $\mathcal{X}$ and $\mathcal{Y}$ are convex, $\boldsymbol{d}$ and $\boldsymbol{W}$ are deterministic and Ξ is described in terms of its extreme points. We close our analysis of the two-stage robust optimization problem (6.1′) with a special case where it reduces to a single-stage robust optimization problem.

Proposition 6.4 (Reduction to Static Problem)**.** *The problem* (6.1′) *reduces to a static robust optimization problem where* $\mathcal{Y}$ *is replaced with its convex hull, if* Ξ *is convex and only the objective function is uncertain, irrespective of* $\mathcal{X}$ *and* $\mathcal{Y}$*.*

Proof. Assume that Ξ is convex and that only the objective function in problem (6.1′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi} \in \Xi$. Problem (6.1′) then simplifies to

$$\inf_{\boldsymbol{x}\in\mathcal{X}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{y}\in\operatorname{conv}\mathcal{Y}} \left\{ \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top\boldsymbol{y} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} \right\},$$

where the replacement of the second-stage feasible region $\mathcal{Y}$ with its convex hull, $\operatorname{conv}\mathcal{Y}$, is justified since the inner minimization has a linear objective function. The classical minimax theorem now allows us to exchange the order of the inner two operators:

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\operatorname{conv}\mathcal{Y}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top\boldsymbol{y} \right\} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} \right\}.$$

This problem is readily recognized as a single-stage robust optimization problem.

We emphasize that the previous argument requires Ξ to be convex. Indeed, we have

$$-1 \;=\; \sup_{\xi\in\{-1,1\}} \inf_{y\in[-1,1]} \xi y \;\neq\; \inf_{y\in[-1,1]} \sup_{\xi\in\{-1,1\}} \xi y \;=\; 0,$$

and we cannot establish equivalence by replacing Ξ with $\operatorname{conv}\Xi = [-1,1]$ in the second optimization problem either. □

A related result was established in [38, Theorem 2.1], where it is shown that the two-stage robust optimization problem (6.1′) reduces to a single-stage robust optimization problem if $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and the uncertain parameters can be partitioned into subsets such that each constraint is only affected by the parameters of one such subset, and no two constraints are affected by parameters of the same subset.

6.2.2 *Analysis of the K-Adaptability Problem*

To analyze problem (6.2), we equivalently rewrite it as

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \boldsymbol{c}^\top\boldsymbol{x} + \mathcal{Q}(\boldsymbol{x},\boldsymbol{y}) \quad \text{with} \qquad \mathcal{Q}(\boldsymbol{x},\boldsymbol{y}) = \sup_{\boldsymbol{\xi}\in\Xi} Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}),$$
$$\text{where} \quad Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}) = \inf_{k\in\mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\} \tag{6.2′}$$

for $\mathcal{Q} : \mathcal{X} \times \mathcal{Y}^K \mapsto \mathbb{R}\cup\{+\infty\}$ and $Q : \mathcal{X} \times \mathcal{Y}^K \times \Xi \mapsto \mathbb{R}\cup\{+\infty\}$.

In analogy to Proposition 6.1, we first investigate whether (6.2′) attains its infima and supremum.

Proposition 6.5 (Continuity). *Problem (6.2′) satisfies the following properties.*

(i) The problem $\mathcal{Q}(\boldsymbol{x},\boldsymbol{y})$ attains its supremum, if only the objective function or only the constraints are uncertain. Otherwise, $\mathcal{Q}(\boldsymbol{x},\boldsymbol{y})$ does not necessarily attain its supremum, even if the constraint left-hand sides are not uncertain.

(ii) The problem (6.2′) attains its infimum, if it is feasible.

Proof. In view of the first statement, assume that only the objective function in problem (6.2′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi}\in\Xi$. Then the inner problem $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})$ simplifies to $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}) = \inf\{\boldsymbol{y}_k^\top \boldsymbol{d}(\boldsymbol{\xi}) \,:\, k\in\mathcal{K}_{\boldsymbol{W}}\}$, where $\mathcal{K}_{\boldsymbol{W}} = \{k\in\mathcal{K} \,:\, \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h}-\boldsymbol{T}\boldsymbol{x}\}$. If $\mathcal{K}_{\boldsymbol{W}} = \emptyset$, then $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}) = +\infty$ for all $\boldsymbol{\xi}\in\Xi$, and any $\boldsymbol{\xi}\in\Xi$ attains the supremum. Otherwise, $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})$ is readily verified to be continuous in $\boldsymbol{\xi}$ for every fixed $\boldsymbol{x}$ and $\boldsymbol{y}$, and [223, Theorem 1.9] then implies that $\mathcal{Q}(\boldsymbol{x},\boldsymbol{y})$ attains its supremum since Ξ is a compact set.

Assume now that only the constraints in problem (6.2′) are uncertain. Then $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})$ simplifies to $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}) = \inf\{\boldsymbol{d}^\top\boldsymbol{y}_k \,:\, k\in\mathcal{K},\ \boldsymbol{\xi}\in\Xi_k\}$, where $\Xi_k = \{\boldsymbol{\xi}\in\Xi \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi})\}$. If $\bigcup_{k\in\mathcal{K}}\Xi_k \neq \Xi$, then $Q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}) = +\infty$ for any $\boldsymbol{\xi}\in\Xi\setminus\bigcup_{k\in\mathcal{K}}\Xi_k$ and

$\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ attains its supremum. Otherwise, $\bigcup_{k \in \mathcal{K}} \Xi_k = \Xi$, and the supremum in $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ is attained by any $\boldsymbol{\xi} \in \bigcap_{k \in \mathcal{K}^\star} \Xi_k$, where

$$\mathcal{K}^\star \in \operatorname*{arg\,max}_{\mathcal{K}' \subseteq \mathcal{K}} \left\{ \min_{k \in \mathcal{K}'} \boldsymbol{d}^\top \boldsymbol{y}_k \,:\, \bigcap_{k \in \mathcal{K}'} \Xi_k \neq \emptyset \right\}.$$

Finally, assume that both the objective function and the constraint right-hand sides in problem (6.2′) are allowed to be uncertain, and consider the following instance of problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$:

$$\sup_{\xi \in [0,1]} \inf_{k \in \{1,2\}} \{\xi - y_k \,:\, y_k \leq \xi\} \qquad \text{with } y_1 = 1 \text{ and } y_2 = 0.$$

The inner minimization problem is optimized by $k^\star = 1$ if $\xi = 1$ and $k^\star = 2$ otherwise. The supremum is 1, and it is approached by the sequence of feasible solutions $\xi \uparrow 1$. Note, however, that the limit point $\xi^\star = 1$ of this sequence results in an objective function value of 0.

As for the second statement, we first note that each extended real-valued function

$$Q_k(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = \begin{cases} \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k & \text{if } \boldsymbol{y}_k \in \mathcal{Y} \text{ and } \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}), \\ +\infty & \text{otherwise,} \end{cases} \qquad k \in \mathcal{K},$$

is lower semicontinuous in $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ since the sets $\{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \in \mathbb{R}^{N_1} \times (\mathbb{R}^{N_2})^K \times \mathbb{R}^{N_p} \,:\, \boldsymbol{y}_k \in \mathcal{Y},\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi})\}$ are closed. From [223, Proposition 1.26] we conclude that the lower semicontinuity is preserved by the partial minimization over k and the partial maximization over $\boldsymbol{\xi}$. Due to [223, Theorem 1.9] the problem (6.2′) then attains its infimum whenever it is feasible. □

Similar to Proposition 6.2, we now consider the convexity properties of problem (6.2′).

Proposition 6.6 (Convexity)**.** *Problem (6.2′) satisfies the following properties.*

(i) The problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ is convex, if Ξ is convex and only the objective function is uncertain. Otherwise, $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ is typically not convex, even if Ξ is convex and only the constraint right-hand sides are uncertain.

(ii) Problem (6.2′) is typically not convex, even if $\mathcal{X}$ and $\mathcal{Y}$ are convex and Ξ is a singleton.

Proof. In view of the first statement, assume that Ξ is convex and that only the objective function in problem (6.2′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi} \in \Xi$. Then the inner problem $Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ simplifies to $Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = \inf\{\boldsymbol{y}_k^\top \boldsymbol{d}(\boldsymbol{\xi}) \,:\, k \in \mathcal{K}_{\boldsymbol{W}}\}$, where $\mathcal{K}_{\boldsymbol{W}} = \{k \in \mathcal{K} \,:\, \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h} - \boldsymbol{T}\boldsymbol{x}\}$. Assume that $\mathcal{K}_{\boldsymbol{W}} \neq \emptyset$; the other case is trivial. Then $Q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ is a piecewise-affine concave function in $\boldsymbol{\xi}$ for every fixed $\boldsymbol{x}$ and $\boldsymbol{y}$. We thus conclude that $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ is a convex optimization problem as it maximizes a concave function over a convex set.

Assume now that the constraint right-hand sides in problem (6.2′) are allowed to be uncertain, and consider the following instance of problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$:

$$\sup_{\xi \in [-1,1]} \inf_{k \in \mathcal{K}} \{y_k \,:\, y_k \geq \xi\} \qquad \text{with } y_1 = 1 \text{ and } y_2 = 0.$$

The inner minimization problem is optimized by $k^\star = 1$, if $\xi > 0$, and $k^\star = 2$, otherwise. The outer maximization problem thus maximizes the non-convex function $\mathbb{I}[\xi > 0]$ over the interval $[-1, 1]$, which amounts to a non-convex optimization problem.

In view of the second statement, consider the following problem instance:

$$\inf_{y_1, y_2 \in [-1,1]} \sup_{\xi \in \{1\}} \inf_{k \in \{1,2\}} \xi \, y_k$$

The problem attains the objective value -1 for $(y_1, y_2) \in \{(-1, 1), (1, -1)\}$, but it attains the larger objective value of 0 for $(y_1, y_2) = 1/2 \cdot (-1, 1) + 1/2 \cdot (1, -1) = (0, 0)$. □

In analogy to Proposition 6.3, we now investigate under which conditions problem (6.2′) is tractable.

Proposition 6.7 (Tractability)**.** *Problem* (6.2′) *satisfies the following properties.*

(i) The problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ can be solved in polynomial time, if Ξ is convex and only the objective function is uncertain. Otherwise, $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ is strongly NP-hard, even if Ξ is convex and only the constraint right-hand sides are uncertain.

(ii) The problem (6.2′) *can be solved in polynomial time, if $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and only the objective function is uncertain. Otherwise, the problem is strongly NP-hard, even if $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and only the constraint right-hand sides are uncertain.*

Proof. The first part of the first statement follows directly from [145, Observation 2].

In view of the second part of the first statement, we consider the following variant of the IP feasibility problem:

> APPROXIMATE 0/1 INTEGER PROGRAMMING FEASIBILITY.
>
> **Instance.** Given are $\boldsymbol{A} \in \mathbb{Z}^{R \times N_p}$, $\boldsymbol{b} \in \mathbb{Z}^R$.
> **Question.** Is there $\boldsymbol{\xi} \in ([0, \epsilon) \cup (1 - \epsilon, 1])^{N_p}$, $\epsilon = \left(\min_r \sum_q |A_{rq}|\right)^{-1}$, such that $\boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b}$?

It follows from [269, Lemma 2] that the approximate IP feasibility problem is strongly NP-hard. We claim that the approximate IP feasibility problem has an affirmative answer if and only if

$$\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y}) \;=\; \sup_{\boldsymbol{\xi} \in \Xi} \inf_{k \in \mathcal{K}} \left\{ y_k^0 \,:\, \underline{\boldsymbol{y}}_k \leq \boldsymbol{\xi} \leq \overline{\boldsymbol{y}}_k \right\} \;=\; 1, \tag{6.4}$$

where $\Xi = \{\boldsymbol{\xi} \in [0,1]^{N_p} \,:\, \boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b}\}$ for $(\boldsymbol{A}, \boldsymbol{b})$ from the approximate IP feasibility instance, $K = N_p + 1$, $\boldsymbol{y}_k = (y_k^0, \underline{\boldsymbol{y}}_k, \overline{\boldsymbol{y}}_k) = (0, \epsilon\, \mathbf{e}_k, \mathbf{e} - \epsilon\, \mathbf{e}_k)$ for $k = 1, \ldots, N_p$ and $\boldsymbol{y}_{N_p+1} =$

$(y^0_{N_p+1}, \underline{\boldsymbol{y}}_{N_p+1}, \overline{\boldsymbol{y}}_{N_p+1}) = (1, \mathbf{0}, \mathbf{e})$. Note that we do not specify the first-stage decision $\boldsymbol{x}$ as it is not required for the argument.

Assume first that the approximate IP feasibility problem has an affirmative answer, i.e., there is $\boldsymbol{\xi}^\star \in ([0,\epsilon) \cup (1-\epsilon, 1])^{N_p}$ such that $\boldsymbol{A}\boldsymbol{\xi}^\star \leq \boldsymbol{b}$. Note that $\boldsymbol{\xi}^\star \in \Xi$, but for every $k = 1, \ldots, N_p$ we have $\boldsymbol{\xi}^\star \notin [\underline{\boldsymbol{y}}_k, \overline{\boldsymbol{y}}_k]$. Since $\boldsymbol{\xi}^\star \in [\mathbf{0}, \mathbf{e}]$, $\boldsymbol{y}_{N_p+1}$ is the only feasible candidate policy, and the optimal value of problem (6.4) is indeed 1.

Assume now that the optimal value of problem (6.4) is 1. In that case, there is $\boldsymbol{\xi}^\star \in \Xi$ such that $\boldsymbol{\xi}^\star \in [\mathbf{0}, \mathbf{e}] \setminus \bigcup_{k=1}^{N_p} [\epsilon\, \mathbf{e}_k, \mathbf{e} - \epsilon\, \mathbf{e}_k]$, i.e, $\boldsymbol{\xi}^\star \in ([0,\epsilon) \cup (1-\epsilon, 1])^{N_p}$. Since $\boldsymbol{A}\boldsymbol{\xi}^\star \leq \boldsymbol{b}$ by construction of Ξ, we conclude that the approximate IP feasibility problem has an affirmative answer.

As for the first part of the second statement, assume first that $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and that only the objective function in problem (6.2′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi} \in \Xi$. The classical minimax theorem then implies that

$$\begin{aligned} &\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}_1\in\mathcal{Y}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_1 \right\} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_1 \leq \boldsymbol{h} \right\} \\ &\quad = \inf_{\boldsymbol{x}\in\mathcal{X}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} \right\}, \end{aligned}$$

i.e., the 1-adaptability problem achieves the same objective value as the two-stage robust optimization problem (6.1). Since the K-adaptability problem is bounded from above by the 1-adaptability problem and from below by the two-stage robust optimization problem, we thus conclude that, under the stated assumptions, the 1-adaptability problem coincides with the K-adaptability problem. The statement now follows from the fact that we can reformulate the 1-adaptability problem as a linear program by dualizing the inner maximization problem, see [36]. Note that the convexity of Ξ is needed in this argument since

$$-1 \;=\; \max_{\xi\in\{-1,1\}} \min_{y\in[-1,1]} \xi y \;\neq\; \min_{y\in[-1,1]} \max_{\xi\in\{-1,1\}} \xi y \;=\; 0.$$

A similar argument with $\Xi = [-1, 1]$ and $\mathcal{Y} = \{-1, 1\}$ shows that the convexity of $\mathcal{Y}$ is needed, too.

Finally, in view of the second part of the second statement, consider the following problem:

$$\inf_{\boldsymbol{y}\in\mathcal{Y}^K} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \{0 \,:\, \boldsymbol{y}_k \leq \boldsymbol{\xi} \leq \mathbf{e} - \boldsymbol{y}_k\}, \tag{6.5}$$

where $K = N_p$, $\mathcal{Y} = \{\boldsymbol{y} \in \mathbb{R}^{N_p}_+ \,:\, \mathbf{e}^\top \boldsymbol{y} = \epsilon\}$, $\epsilon < 1/2$, and $\Xi = \{\boldsymbol{\xi} \in [0,1]^{N_p} \,:\, \boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b}\}$ for $(\boldsymbol{A}, \boldsymbol{b})$ from an approximate IP feasibility instance. We claim that the optimal value of this problem is $+\infty$, i.e., there is $\boldsymbol{\xi} \in \Xi$ for which no decision $\boldsymbol{y} \in \mathcal{Y}^K$ is feasible in the second stage, if and only if the approximate IP feasibility instance has an affirmative answer.

Assume first that the approximate IP feasibility problem has an affirmative answer. It then follows from [269, Lemma 2] that the *exact* IP feasibility problem also has an affirmative answer; that is, there is $\boldsymbol{\xi}^\star \in \{0,1\}^{N_p}$ such that $\boldsymbol{A}\boldsymbol{\xi}^\star \leq \boldsymbol{b}$. Fix any feasible candidate policy $\boldsymbol{y}_k \in \mathcal{Y}$. By construction, there is $i \in \{1,\dots,N_p\}$ such that $y_{ki} > 0$. Thus, the candidate policy is feasible in the second stage under the parameter realization $\boldsymbol{\xi}^\star$ only if $\xi_i^\star \in [y_{ki}, 1 - y_{ki}] \subseteq (0,1)$, which is not the case since $\xi_i^\star \in \{0,1\}$. We thus conclude that the optimal value of problem (6.5) is indeed $+\infty$ whenever the approximate IP feasibility problem has an affirmative answer.

Assume now that the optimal value of problem (6.5) is $+\infty$ and consider the K candidate policies $\boldsymbol{y}_k = \epsilon\, \mathbf{e}_k$, $k = 1,\dots,N_p$. Since $\boldsymbol{y} = (\boldsymbol{y}_1,\dots,\boldsymbol{y}_K) \in \mathcal{Y}^K$ and problem (6.5) evaluates to $+\infty$ under this choice of $\boldsymbol{y}$, there must be $\boldsymbol{\xi}^\star \in \Xi$ such that $\boldsymbol{\xi}^\star \in [\mathbf{0}, \mathbf{e}] \setminus \bigcup_{k=1}^{N_p} [\epsilon\, \mathbf{e}_k, \mathbf{e} - \epsilon \mathbf{e}_k]$; that is, $\boldsymbol{\xi}^\star \in ([0,\epsilon) \cup (1-\epsilon, 1])^{N_p}$. Since $\boldsymbol{A}\boldsymbol{\xi}^\star \leq \boldsymbol{b}$ by construction of Ξ, we conclude that the approximate IP feasibility problem has an affirmative answer. □

We note that $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ can be solved in polynomial time if Ξ is convex and the number of policies K is fixed (even when the objective function, the constraint coefficients and the right-hand sides are uncertain), see [145, Corollary 1]. The NP-hardness of $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{y})$ has previously been established under the more restrictive assumption that both the objective function and the constraint right-hand sides in problem (6.2′) are uncertain, see [145, Theorem 3]. For the special case where $K = 2$, the K-adaptability problem with objective and constraint uncertainty can be solved in polynomial time if any of N_p, $\max\{N_1, N_2\}$ or L is fixed [43, Proposition 5], while the problem becomes NP-hard otherwise [43, Proposition 6]. Proposition 6.7 (ii) provides an alternative proof of the NP-hardness of problem (6.2′), which facilitates a direct comparison to the two-stage robust optimization problem (6.1′), see Proposition 6.3 (iii).

We also note that the K-adaptability problem (6.2′) simplifies in the absence of first-stage decisions $\boldsymbol{x}$. For this case, it has been shown in [69, Theorem 2] that the problem can be solved in polynomial time, if only the objective function is uncertain, the deterministic second-stage problem is polynomial time solvable, the uncertainty set Ξ has a tractable representation, and if any number of policies $K > N_2$ is acceptable. The same problem becomes NP-hard, however, when the number of policies K is fixed [69, Corollary 3] or when the uncertainty sets are discrete [70, Corollaries 1–4].

Similar to Proposition 6.4, we next consider when the K-adaptability problem (6.2′) reduces to a single-stage robust optimization problem.

Proposition 6.8 (Reduction to Static Problem)**.** *The problem* (6.2′) *reduces to a static robust optimization problem where $\mathcal{Y}$ is replaced with its convex hull, if Ξ is convex, only the objective function is uncertain and $K > \min\{N_2, N_p\}$, irrespective of $\mathcal{X}$ and $\mathcal{Y}$.*

Proof. Assume that Ξ is convex and that only the objective function in problem (6.2′) is uncertain; that is, $\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{T}$, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{h}$ for all $\boldsymbol{\xi} \in \Xi$. Problem (6.2′) then simplifies to

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h} \right\}.$$

The inner discrete minimization can be replaced by a continuous minimization over all convex combinations $\boldsymbol{\lambda} \in \Delta(\boldsymbol{x}, \boldsymbol{y}) = \left\{ \boldsymbol{\lambda} \in \mathbb{R}^K_+ \,:\, \mathbf{e}^\top \boldsymbol{\lambda} = 1,\ \ \lambda_k = 0 \text{ if } \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k \not\leq \boldsymbol{h} \right\}$:

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{\lambda}\in\Delta(\boldsymbol{x},\boldsymbol{y})} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \sum_{k\in\mathcal{K}} \lambda_k \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \right\}$$

The classical minimax theorem then allows us to exchange the order of the inner two operators:

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \inf_{\boldsymbol{\lambda}\in\Delta(\boldsymbol{x},\boldsymbol{y})} \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \left[\sum_{k\in\mathcal{K}} \lambda_k \boldsymbol{y}_k \right] \right\} \tag{6.6}$$

If $K \geq N_2 + 1$, then we can use Carathéodory's theorem to replace the convex combinations of K candidate decisions $\boldsymbol{y}_k \in \mathcal{Y}$ with a single decision from the convex hull, conv $\mathcal{Y}$, resulting in

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\operatorname{conv}\mathcal{Y}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} \right\} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} \right\}, \tag{6.7}$$

which is readily recognized as a single-stage robust optimization problem. Similarly, problem (6.6) can be rewritten as

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \inf_{\boldsymbol{\lambda}\in\Delta(\boldsymbol{x},\boldsymbol{y})} \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \sum_{k\in\mathcal{K}} \lambda_k \left[\boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \right] \right\},$$

and if $K \geq N_p + 1$, then we can use Carathéodory's theorem to replace the convex combinations of K candidate decisions $\boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \in \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} : \boldsymbol{y} \in \mathcal{Y} \right\}$ with a single decision from the convex hull of their domain, conv $\left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} : \boldsymbol{y} \in \mathcal{Y} \right\} = \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} : \boldsymbol{y} \in \operatorname{conv} \mathcal{Y} \right\}$, resulting again in the single-stage robust optimization problem (6.7).

As in Proposition 6.4, the previous arguments require Ξ to be convex. Indeed, we have

$$-1 \;=\; \sup_{\xi\in\{-1,1\}} \inf_{k\in\{1,2\}} \xi y_k \;\neq\; \inf_{y\in[-1,1]} \sup_{\xi\in\{-1,1\}} \xi y \;=\; 0 \qquad \text{with } y_1 = -1 \text{ and } y_2 = 1,$$

and we cannot establish equivalence by replacing Ξ with conv $\Xi = [-1, 1]$ in the second optimization problem either. To see that the number of policies K must exceed $\min\{N_2, N_p\}$ in general in the previous arguments, we compare the two problems

$$\inf_{\boldsymbol{y}\in(\{-1,1\}^{N_2})^K} \sup_{\boldsymbol{\xi}\in[-1,1]^{N_p}} \inf_{k\in\mathcal{K}} \xi_1 y_{k1} \qquad \text{and} \qquad \inf_{\boldsymbol{y}\in[-1,1]^{N_2}} \sup_{\boldsymbol{\xi}\in[-1,1]^{N_p}} \xi_1 y_1.$$

For every $N_2, N_p \geq 1$, the problem on the right attains its optimal value 0 at any $\boldsymbol{y} \in [-1,1]^{N_2}$ with $y_1 = 0$. Similarly, for every $N_2, N_p \geq 1$, the problem on the left attains an optimal value of 0, if $K \geq 2$, and an optimal value of 1, if $K = 1$. We thus verify that $K > \min\{N_2, N_p\}$ is required for the optimal values to coincide by choosing $(N_2, N_p) \in \{(1,2),(2,1)\}$. □

We close this section with several special cases where the optimal value of the K-adaptability problem (6.2′) coincides with the optimal value of the two-stage robust optimization problem (6.1′).

Proposition 6.9 (Optimality)**.** *The optimal values of problems* (6.1′) *and* (6.2′) *coincide if* (i) *$\mathcal{Y}$ and Ξ are convex and only the objective function is uncertain, irrespective of $\mathcal{X}$ and K, or if* (ii) *Ξ is convex, only the objective function is uncertain and $K > \min\{N_2, N_p\}$, irrespective of $\mathcal{X}$ and $\mathcal{Y}$, or if* (iii) *$\mathcal{Y}$ has a finite cardinality and $K \geq |\mathcal{Y}|$, irrespective of $\mathcal{X}$ and Ξ. Otherwise, their optimal values may differ for any finite K, even if $\mathcal{X}$, $\mathcal{Y}$ and Ξ are convex and only the constraint right-hand sides are uncertain.*

Proof. The fact that the two optimal values coincide under the first set of conditions follows from the proof of Proposition 6.7 (ii), where we have shown that, under the stated assumptions, the optimal values of the 1-adaptability problem and the two-stage robust optimization problem (6.1′) coincide. We have also shown there that the convexity of $\mathcal{Y}$ and Ξ is crucial for the proof to hold.

The fact that the two optimal values coincide under the second set of conditions follows from Proposition 6.8, which shows that, under the stated assumptions, the K-adaptability problem (6.2′) is equivalent to

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\operatorname{conv}\mathcal{Y}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} \right\} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} \right\}.$$

The classical minimax theorem, which is applicable since $\operatorname{conv}\mathcal{Y}$ and Ξ are convex, then implies that this problem is equivalent to

$$\inf_{\boldsymbol{x}\in\mathcal{X}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{y}\in\operatorname{conv}\mathcal{Y}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} \,:\, \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{h} \right\},$$

which provides a lower bound to the two-stage robust optimization problem (6.1′) since $\operatorname{conv}\mathcal{Y} \supseteq \mathcal{Y}$. On the other hand, we know that the K-adaptability problem (6.2′) by construction bounds (6.1′) from above. We thus conclude that the optimal values of both problems must coincide.

In view of the third set of conditions, it is clear that $K \geq |\mathcal{Y}|$ policies are sufficient for the optimal values of problems (6.1′) and (6.2′) to coincide. Moreover, [145, Theorem 4] presents a problem where the optimal values of problems (6.1′) and (6.2′) differ for every $K < |\mathcal{Y}|$.

As for the second part of the statement, consider the problem

$$\sup_{\xi\in[-1,1]} \inf_{\substack{\tau\in\mathbb{R},\\ y\in[-1,1]}} \{\tau : \tau \geq \xi - y,\ \ \tau \geq y - \xi\}.$$

The optimal second-stage decision to this two-stage robust optimization problem satisfies $\tau(\xi) = 0$ and $y(\xi) = \xi$, and any $\xi \in [-1,1]$ attains the optimal objective value 0. Consider now the K-adaptability problem

$$\inf_{\substack{\boldsymbol{\tau}\in\mathbb{R}^K,\\ \boldsymbol{y}\in[-1,1]^K}} \sup_{\xi\in[-1,1]} \inf_{k\in\mathcal{K}} \{\tau_k : \tau_k \geq \xi - y_k,\ \ \tau_k \geq y_k - \xi\}.$$

The objective value of this problem evaluates to at least $\max_{\xi\in[-1,1]} \min_{k\in\mathcal{K}} |\xi - y_k| > 0$ for any finite K and any feasible solution $(\boldsymbol{\tau}, \boldsymbol{y}) \in \mathbb{R}^K \times [-1,1]^K$. □

The validity of the first part of the statement under the second set of conditions in Proposition 6.9 was established for the subclass of purely binary K-adaptability problems in [145, Theorem 1]. We also mention that the optimal value of the K-adaptability problem (6.2′) approaches the optimal value of the two-stage robust optimization problem (6.1′) if $K \to \infty$ and a continuity assumption is satisfied, see [43, Proposition 1].

6.2.3 *Incorporating Decision Rules in the K-Adaptability Problem*

Although the K-adaptability problem (6.2) selects the best candidate policy $\boldsymbol{y}_k$ in response to the observed parameter realization $\boldsymbol{\xi} \in \Xi$, the policies $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K$—once selected in the first stage—no longer depend on $\boldsymbol{\xi}$. This lack of dependence on the uncertain problem parameters can lead to overly conservative approximations of the two-stage robust optimization problem (6.1) when the second-stage decisions are continuous. In this section, we show how the K-adaptability problem (6.2) can be used to generalize affine decision rules, which are commonly used to approximate continuous instances of the two-stage robust optimization problem (6.1). We note that existing schemes, such as [69, 145], cannot be used for this purpose as they require the wait-and-see decisions $\boldsymbol{y}$ to be binary.

Throughout this section, we assume that problem (6.1) has purely continuous second-stage decisions (that is, $\mathcal{Y}$ is LP representable), a deterministic objective function (that is, $\boldsymbol{d}(\boldsymbol{\xi}) = \boldsymbol{d}$ for all $\boldsymbol{\xi} \in \Xi$) and fixed recourse (that is, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ for all $\boldsymbol{\xi} \in \Xi$). The assumption of continuous second-stage decisions allows us to assume, without loss of generality, that $\mathcal{Y} = \mathbb{R}^{N_2}$ as any potential restrictions can be absorbed in the second-stage constraints.

The affine decision rule approximation to the two-stage robust optimization problem is

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}:\Xi\overset{1}{\mapsto}\mathbb{R}^{N_2}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{d}^\top\boldsymbol{y}(\boldsymbol{\xi}) \right\} : \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}(\boldsymbol{\xi}) \leq \boldsymbol{h}(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} \in \Xi \right\},$$

where $\boldsymbol{y} : \Xi \overset{1}{\mapsto} \mathbb{R}^{N_2}$ indicates that $\boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{y}^0 + \boldsymbol{Y}\boldsymbol{\xi}$ for some $\boldsymbol{y}^0 \in \mathbb{R}^{N_2}$ and $\boldsymbol{Y} \in \mathbb{R}^{N_2 \times N_p}$, see [38]. This problem provides a conservative approximation to the two-stage robust optimization problem (6.1) since we replace the space of all (possibly non-convex and discontinuous) second-stage policies $\boldsymbol{y} : \Xi \mapsto \mathbb{R}^{N_2}$ with the subspace of all affine second-stage policies $\boldsymbol{y} : \Xi \overset{1}{\mapsto} \mathbb{R}^{N_2}$. In a similar spirit, we define the subspace of all piecewise affine decision rules $\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}$ with K pieces as

$$\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2} \quad \Longleftrightarrow \quad \left[\begin{array}{l} \exists (\boldsymbol{y}_k^0, \boldsymbol{Y}_k) \in \mathbb{R}^{N_2} \times \mathbb{R}^{N_2 \times N_p},\ k = 1, \ldots, K, \text{ such that} \\ \forall \boldsymbol{\xi} \in \Xi,\ \exists k \in \{1, \ldots, K\} \,:\, \boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{y}_k^0 + \boldsymbol{Y}_k \boldsymbol{\xi} \end{array} \right].$$

Note that our earlier definition of $\boldsymbol{y} : \Xi \overset{1}{\mapsto} \mathbb{R}^{N_2}$ is identical to our definition of $\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}$ if $K = 1$. For $K > 1$, the decision rules $\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}$ may be non-convex and discontinuous, and the regions where $\boldsymbol{y}$ is affine may be non-closed and non-convex. We highlight that the points of nonlinearity are determined by the optimization problem. This is in contrast to many existing solution schemes for piecewise affine decision rules, such as [47, 79, 128, 131], where these points are specified ad hoc by the decision-maker.

Observation 6.1. *The piecewise affine decision rule problem with fixed recourse*

$$\inf_{\substack{\boldsymbol{x} \in \mathcal{X}, \\ \boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}}} \left\{ \sup_{\boldsymbol{\xi} \in \Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y}(\boldsymbol{\xi}) \right\} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}(\boldsymbol{\xi}) \leq \boldsymbol{h}(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} \in \Xi \right\} \tag{6.8}$$

is equivalent to the K-adaptability problem with random recourse

$$\inf_{\substack{\boldsymbol{x} \in \mathcal{X}, \\ (\boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z}) \in \hat{\mathcal{Y}}^K}} \sup_{\boldsymbol{\xi} \in \Xi} \inf_{k \in \mathcal{K}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y}_k^0 + \hat{\boldsymbol{d}}(\boldsymbol{\xi})^\top \boldsymbol{z}_{k1} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \hat{\boldsymbol{W}}(\boldsymbol{\xi})\boldsymbol{z}_{k2} \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \tag{6.9}$$

where $\hat{\mathcal{Y}} = \left\{ (\boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z}) \in \mathbb{R}^{N_2} \times \mathbb{R}^{N_2 \times N_p} \times (\mathbb{R}^{N_p} \times \mathbb{R}^{N_p L}) \,:\, \boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2) \text{ with } \boldsymbol{z}_1 = \boldsymbol{Y}^\top \boldsymbol{d} \text{ and } \boldsymbol{z}_2 = [\boldsymbol{w}_1^\top \boldsymbol{Y} \ \ldots \ \boldsymbol{w}_L^\top \boldsymbol{Y}]^\top \right\}$, $\hat{\boldsymbol{d}}(\boldsymbol{\xi}) = \boldsymbol{\xi}$ *and* $\hat{\boldsymbol{W}}(\boldsymbol{\xi}) = \operatorname{diag}(\boldsymbol{\xi}^\top, \ldots, \boldsymbol{\xi}^\top) \in \mathbb{R}^{L \times N_p L}$.

Proof. Problem (6.9) is infeasible if and only if for every $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z}) \in \hat{\mathcal{Y}}^K$ there is a $\boldsymbol{\xi} \in \Xi$ such that $\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \hat{\boldsymbol{W}}(\boldsymbol{\xi})\boldsymbol{z}_{k2} \not\leq \boldsymbol{h}(\boldsymbol{\xi})$ for all $k = 1, \ldots, K$, which in turn is the case if and only if for every $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}_k^0, \boldsymbol{Y}_k) \in \mathbb{R}^{N_2} \times \mathbb{R}^{N_2 \times N_p}$, $k = 1, \ldots, K$ there is a $\boldsymbol{\xi} \in \Xi$ such that $\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \boldsymbol{W}\boldsymbol{Y}_k\boldsymbol{\xi} \not\leq \boldsymbol{h}(\boldsymbol{\xi})$ for all $k = 1, \ldots, K$; that is, if and only if problem (6.8) is infeasible. We thus assume that both (6.8) and (6.9) are feasible. In this case, we verify that every feasible solution $(\boldsymbol{x}, \boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z})$ to problem (6.9) gives rise to a feasible solution $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{y}_{k(\boldsymbol{\xi})}^0 + \boldsymbol{Y}_{k(\boldsymbol{\xi})}\boldsymbol{\xi}$ and $k(\boldsymbol{\xi})$ is any element of $\arg\min_{k \in \mathcal{K}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y}_k^0 + \hat{\boldsymbol{d}}(\boldsymbol{\xi})^\top \boldsymbol{z}_{k1} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \hat{\boldsymbol{W}}(\boldsymbol{\xi})\boldsymbol{z}_{k2} \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}$, in problem (6.8) that attains the same worst-case objective value. Similarly, every *optimal solution* $(\boldsymbol{x}, \boldsymbol{y})$ to problem (6.8) gives rise to an optimal solution $(\boldsymbol{x}, \boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z})$, where $\boldsymbol{z}_k = (\boldsymbol{z}_{k1}, \boldsymbol{z}_{k2})$ with

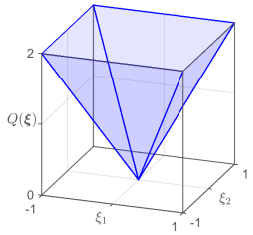


Figure 6.1: The optimal second-stage value function in Example 6.1 is given by the first-order cone $Q^\star(\boldsymbol{\xi}) = [\xi_1 + \xi_2]_+ + [\xi_1 - \xi_2]_+ + [-\xi_1 + \xi_2]_+ + [-\xi_1 - \xi_2]_+$.

$\boldsymbol{z}_{k1} = \boldsymbol{Y}_k^\top \boldsymbol{d}$ and $\boldsymbol{z}_{k2} = [\boldsymbol{w}_1^\top \boldsymbol{Y}_k \ \ldots \ \boldsymbol{w}_L^\top \boldsymbol{Y}_k]^\top$, $k = 1, \ldots, K$, in problem (6.9). Hence, (6.8) and (6.9) share the same optimal value and the same sets of optimal solutions. □

We close with an example that illustrates the benefits of piecewise affine decision rules.

Example 6.1. *Consider the following instance of the two-stage robust optimization problem (6.1), which has been proposed in [136, Section 5.2]:*

$$\sup_{\boldsymbol{\xi} \in [-1,1]^2} \inf_{\boldsymbol{y} \in \mathbb{R}^4_+} \left\{ \boldsymbol{e}^\top \boldsymbol{y} \,:\, y_1 \geq \xi_1 + \xi_2,\ y_2 \geq \xi_1 - \xi_2,\ y_3 \geq -\xi_1 + \xi_2,\ y_4 \geq -\xi_1 - \xi_2 \right\}.$$

The optimal second-stage policy *is* $\boldsymbol{y}^\star(\boldsymbol{\xi}) = ([\xi_1 + \xi_2]_+, [\xi_1 - \xi_2]_+, [-\xi_1 + \xi_2]_+, [-\xi_1 - \xi_2]_+)$, *where* $[\cdot]_+ = \max\{\cdot, 0\}$, *and it results in the optimal second-stage value function* $Q^\star(\boldsymbol{\xi}) = [\xi_1 + \xi_2]_+ + [\xi_1 - \xi_2]_+ + [-\xi_1 + \xi_2]_+ + [-\xi_1 - \xi_2]_+$ *with a worst-case objective value of* 2, *see Figure 6.1. The* best affine decision rule $\boldsymbol{y}^1 : \Xi \overset{1}{\mapsto} \mathbb{R}^4_+$ *is* $\boldsymbol{y}^1(\boldsymbol{\xi}) = (1 + \xi_2, 1 + \xi_1, 1 - \xi_1, 1 - \xi_2)$, *and it results in the constant second-stage value function* $Q^1(\boldsymbol{\xi}) = 4$. *The* best 2-adaptable affine decision rule $\boldsymbol{y}^2 : \Xi \overset{2}{\mapsto} \mathbb{R}^4_+$, *on the other hand, is given by*

$$\boldsymbol{y}^2(\boldsymbol{\xi}) = \begin{cases} (0,\ 1 + \xi_1,\ 1 + \xi_2,\ -\xi_1 - \xi_2) & \text{if } \xi_1 + \xi_2 \leq 0 \\ (\xi_1 + \xi_2,\ 1 - \xi_2,\ 1 - \xi_1,\ 0) & \text{otherwise,} \end{cases}$$

and it results in the constant second-stage value function $Q^2(\boldsymbol{\xi}) = 2$. *Thus, 2-adaptable affine decision rules are optimal in this example. Figure 6.2 illustrates the optimal value, the affine approximation and the 2-adaptable affine approximation of the decision variable* y_3.

The piecewise affine decision rules presented here can be readily combined with discrete second-stage decisions. For the sake of brevity, we omit the details of this straightforward extension.

6.3 SOLUTION SCHEME

Our solution scheme for the K-adaptability problem (6.2) is based on a reformulation as a semi-infinite disjunctive program which we present next.

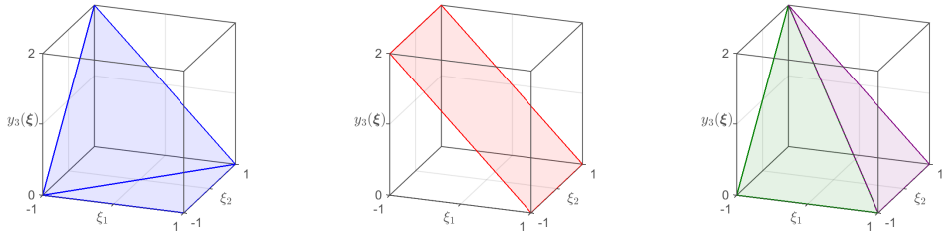


Figure 6.2: Plot of the optimal second-stage policy $y_3(\xi)$ in the two-stage robust optimization problem (left), the affine decision rule problem (middle) and the 2-adaptable affine decision rule problem (right).

Observation 6.2. *The K-adaptability problem* (6.2) *is equivalent to*

$$\begin{array}{ll} \text{minimize} & \theta \\ \text{subject to} & \theta \in \mathbb{R},\ \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} \in \mathcal{Y}^K \\ & \displaystyle\bigvee_{k\in\mathcal{K}} \begin{bmatrix} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \end{bmatrix} \quad \forall \boldsymbol{\xi} \in \Xi. \end{array} \tag{6.10}$$

Moreover, if some of the constraints in problem (6.10) *are deterministic, i.e., they do not depend on* $\boldsymbol{\xi}$*, then they can be moved outside the disjunction and instead be enforced for all* $k \in \mathcal{K}$*.*

In the following, we stipulate that the optimal value of (6.10) is $+\infty$ whenever it is infeasible.

Proof of Observation 6.2. Problem (6.2) is infeasible if and only if (iff) for every $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}^K$ there is a $\boldsymbol{\xi} \in \Xi$ such that $\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \not\leq \boldsymbol{h}(\boldsymbol{\xi})$ for all $k \in \mathcal{K}$, which in turn is the case iff for every $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}^K$, the disjunction in (6.10) is violated for at least one $\boldsymbol{\xi} \in \Xi$; that is, iff problem (6.10) is infeasible. We thus assume that both (6.2) and (6.10) are feasible. In this case, one readily verifies that every feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ to problem (6.2) gives rise to a feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$, where $\theta = \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \{\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k : \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi})\}$, in problem (6.10) with the same objective value. Likewise, any *optimal* solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to problem (6.10) corresponds to an optimal solution $(\boldsymbol{x}, \boldsymbol{y})$ in problem (6.2). Hence, (6.2) and (6.10) share the same optimal value and the same sets of optimal solutions.

We now claim that if $\boldsymbol{t}_l(\boldsymbol{\xi})^\top = \boldsymbol{t}_l^\top$, $\boldsymbol{w}_l(\boldsymbol{\xi})^\top = \boldsymbol{w}_l^\top$ and $h_l(\boldsymbol{\xi}) = h_l$ for all $l \in \{1, \ldots, L\} \setminus \mathcal{L}$, where $\mathcal{L} \subseteq \{1, \ldots, L\}$, then problem (6.10) is equivalent to

$$\begin{array}{lll} \text{minimize} & \theta & \\ \text{subject to} & \theta \in \mathbb{R},\ \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} \in \mathcal{Y}^K & \\ & \boldsymbol{t}_l^\top \boldsymbol{x} + \boldsymbol{w}_l^\top \boldsymbol{y}_k \leq h_l & \forall l \in \{1, \ldots, L\} \setminus \mathcal{L}, \forall k \in \mathcal{K} \\ & \displaystyle\bigvee_{k\in\mathcal{K}} \begin{bmatrix} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \\ \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq h_l(\boldsymbol{\xi}) \quad \forall l \in \mathcal{L} \end{bmatrix} & \forall \boldsymbol{\xi} \in \Xi. \end{array}$$

By construction, every feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to the above problem is feasible in problem (6.10). Conversely, fix any feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to problem (6.10) and assume that $\boldsymbol{t}_l^\top \boldsymbol{x} + \boldsymbol{w}_l^\top \boldsymbol{y}_k > h_l$ for some $k \in \mathcal{K}$ and $l \in \{1, \ldots, L\} \setminus \mathcal{L}$. In that case, the k^{th} disjunct in (6.10) is violated for *every* realization $\boldsymbol{\xi} \in \Xi$. We can therefore replace $\boldsymbol{y}_k$ with a different candidate policy $\boldsymbol{y}_{k'}$ that satisfies $\boldsymbol{t}_l^\top \boldsymbol{x} + \boldsymbol{w}_l^\top \boldsymbol{y}_{k'} \leq h_l$ for all $l \in \{1, \ldots, L\} \setminus \mathcal{L}$ without sacrificing feasibility. (Note that such a candidate policy $\boldsymbol{y}_{k'}$ exists since $(\theta, \boldsymbol{x}, \boldsymbol{y})$ is assumed to be feasible in (6.10).) Replacing any infeasible policy in this way results in a solution that is feasible in the above problem. □

Problem (6.10) cannot be solved directly as it contains infinitely many disjunctive constraints. Instead, our solution scheme iteratively solves a sequence of (increasingly tighter) relaxations of this problem that are obtained by enforcing the disjunctive constraints over finite subsets of Ξ. Whenever the solution of such a relaxation violates the disjunction for some realization $\boldsymbol{\xi} \in \Xi$, we create K subproblems that enforce the disjunction associated with $\boldsymbol{\xi}$ to be satisfied by the k^{th} disjunct, $k = 1, \ldots, K$. Our solution scheme is reminiscent of discretization methods employed in *semi-infinite programming*, which iteratively replace an infinite set of constraints with finite subsets and solve the resulting discretized problems. Indeed, our scheme can be interpreted as a generalization of Kelley's cutting-plane method [59, 168] applied to semi-infinite disjunctive programs. In the special case where $K = 1$, our method reduces to the cutting-plane method for (static) robust optimization problems proposed in [203].

In the remainder of this section, we describe our basic branch-and-bound scheme (Section 6.3.1), we study its convergence (Section 6.3.2), we discuss algorithmic variants to the basic scheme that can enhance its numerical performance (Section 6.3.3), and we present a heuristic variant that can address problems of larger scale (Section 6.3.4).

6.3.1 *Branch-and-Bound Algorithm*

Our solution scheme iteratively solves a sequence of scenario-based K-adaptability problems and separation problems. We define both problems first, and then we describe the overall algorithm.

THE SCENARIO-BASED K-ADAPTABILITY PROBLEM. For a collection $\Xi_1, \ldots, \Xi_K$ of finite subsets of the uncertainty set Ξ, we define the *scenario-based K-adaptability problem* as

$$\begin{array}{rcll} \mathcal{M}(\Xi_1, \ldots, \Xi_K) & = & \text{minimize} & \theta & \\ & & \text{subject to} & \theta \in \mathbb{R}, \;\; \boldsymbol{x} \in \mathcal{X}, \;\; \boldsymbol{y} \in \mathcal{Y}^K & \\ & & & \left.\begin{array}{l} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \end{array}\right\} & \forall \boldsymbol{\xi} \in \Xi_k, \; \forall k \in \mathcal{K}. \end{array} \tag{6.11}$$

If $\mathcal{X}$ and $\mathcal{Y}$ are convex, problem (6.11) is an LP; otherwise, it is an MILP. The problem is closely related to a relaxation of the semi-infinite disjunctive program (6.10) that enforces the disjunction only over the realizations $\boldsymbol{\xi} \in \bigcup_{k\in\mathcal{K}} \Xi_k$. More precisely, problem (6.11) can be interpreted as a restriction of that relaxation which requires the k^{th} candidate policy $\boldsymbol{y}_k$ to be worst-case optimal for all realizations $\boldsymbol{\xi} \in \Xi_k$, $k \in \mathcal{K}$. We obtain an optimal solution $(\theta, \boldsymbol{x}, \boldsymbol{y}_k)$ to the relaxed semi-infinite disjunctive program by solving $\mathcal{M}(\Xi_1, \ldots, \Xi_K)$ for all partitions $(\Xi_1, \ldots, \Xi_K)$ of $\bigcup_{k\in\mathcal{K}} \Xi_k$ and reporting the optimal solution $(\theta, \boldsymbol{x}, \boldsymbol{y}_k)$ of the problem $\mathcal{M}(\Xi_1, \ldots, \Xi_K)$ with the smallest objective value.

If $\Xi_k = \emptyset$ for all $k \in \mathcal{K}$, then problem (6.11) is unbounded, and we stipulate that its optimal value is $-\infty$ and that its optimal value is attained by any solution $(-\infty, \boldsymbol{x}, \boldsymbol{y})$ with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}^K$. Otherwise, if problem (6.11) is infeasible for $\Xi_1, \ldots, \Xi_K$, then we define its optimal value to be $+\infty$. In all other cases, the optimal value of problem (6.11) is finite and it is attained by an optimal solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ since $\mathcal{X}$ and $\mathcal{Y}$ are compact.

Remark 6.2 (Decomposability)**.** *For K-adaptability problems without first-stage decisions $\boldsymbol{x}$, problem (6.11) decomposes into K scenario-based static robust optimization problems that are only coupled through the constraints referencing the epigraph variable θ. In this case, we can recover an optimal solution to problem (6.11) by solving each of the K static problems individually and identifying the optimal θ as the maximum of their optimal values.*

THE SEPARATION PROBLEM. For a feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to the scenario-based K-adaptability problem (6.11), we define the *separation problem* as

$$\begin{aligned} \mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) &= \max_{\boldsymbol{\xi}\in\Xi} \; S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}), \text{ where} \\ S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) &= \min_{k\in\mathcal{K}} \; \max \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta, \; \max_{l\in\{1,\ldots,L\}} \left\{ \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi}) \right\} \right\}, \end{aligned} \tag{6.12}$$

for $\mathcal{S} : \mathbb{R} \cup \{-\infty\} \times \mathcal{X} \times \mathcal{Y}^K \mapsto \mathbb{R} \cup \{+\infty\}$ and $S : \mathbb{R} \cup \{-\infty\} \times \mathcal{X} \times \mathcal{Y}^K \times \Xi \mapsto \mathbb{R} \cup \{+\infty\}$. Whenever it is positive, the innermost maximum in the definition of $S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ records the maximum constraint violation of the candidate policy $\boldsymbol{y}_k$ under the parameter realization $\boldsymbol{\xi} \in \Xi$. Likewise, the quantity $\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta$ denotes the excess of the objective value of $\boldsymbol{y}_k$ under the realization $\boldsymbol{\xi}$ over the current candidate value of the worst-case objective, θ. Thus, $S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ is strictly positive if and only if every candidate policy $\boldsymbol{y}_k$ either is infeasible or results in an objective value greater than θ under the realization $\boldsymbol{\xi} \in \Xi$. Whenever θ is finite, the separation problem is feasible and bounded, and it has an optimal solution since Ξ is nonempty and compact. Otherwise, we have $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) = +\infty$, and the optimal value is attained by any $\boldsymbol{\xi} \in \Xi$.

Observation 6.3. *The separation problem (6.12) is equivalent to the MILP*

$$
\begin{array}{lll}
\text{maximize} & \zeta & \\
\text{subject to} & \zeta \in \mathbb{R},\ \boldsymbol{\xi} \in \Xi,\ z_{kl} \in \{0,1\},\ (k,l) \in \mathcal{K} \times \{0,1,\dots,L\} & \\
& \left.\begin{array}{l}
\displaystyle\sum_{l=0}^{L} z_{kl} = 1 \\
z_{k0} = 1 \ \Rightarrow\ \zeta \le \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta \\
z_{kl} = 1 \ \Rightarrow\ \zeta \le \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi}) \quad \forall l \in \{1,\dots,L\}
\end{array}\right\} \ \forall k \in \mathcal{K}.
\end{array}
\tag{6.13}
$$

This problem can be solved in polynomial time if Ξ is convex and the number of policies K is fixed.

Proof. Fix any feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to the scenario-based K-adaptability problem (6.11). For every $\boldsymbol{\xi} \in \Xi$, we can construct a feasible solution $(\zeta, \boldsymbol{\xi}, \boldsymbol{z})$ to problem (6.13) with $\zeta = S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ by setting $z_{k0} = 1$ if $\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta \ge \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi})$ for all $l = 1,\dots,L$ and $z_{kl} = 1$ for $l \in \operatorname{arg\,max}_{l \in \{1,\dots,L\}} \{\boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi})\}$ otherwise (where ties can be broken arbitrarily). We thus conclude that $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y})$ is less than or equal to the optimal value of problem (6.13). Likewise, every feasible solution $(\zeta, \boldsymbol{\xi}, \boldsymbol{z})$ to problem (6.13) satisfies $\zeta \le \max\{\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta,\ \max_{l \in \{1,\dots,L\}} \{\boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi})\}\}$ for all $k \in \mathcal{K}$; that is, $\zeta \le S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$. Thus, the optimal value of problem (6.13) is less than or equal to $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y})$ as well.

If the number of policies K is fixed and the uncertainty set Ξ is convex, then problem (6.13) can be solved by enumerating all $(L+1)^K$ possible choices for $\boldsymbol{z}$, solving the resulting linear programs in ζ and $\boldsymbol{\xi}$ and reporting the solution with the maximum value of ζ. □

THE ALGORITHM. Our solution scheme solves a sequence of scenario-based K-adaptability problems (6.11) over monotonically increasing scenario sets Ξ_k, $k \in \mathcal{K}$. At each iteration, the separation problem (6.13) identifies a new scenario $\boldsymbol{\xi} \in \Xi$ to be added to these sets.

1. *Initialize.* Set $\mathcal{N} \leftarrow \{\tau^0\}$ (node set), where $\tau^0 = (\Xi_1^0, \dots, \Xi_K^0)$ with $\Xi_k^0 = \emptyset$ for all $k \in \mathcal{K}$ (root node). Set $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}}) \leftarrow (+\infty, \emptyset, \emptyset)$ (incumbent solution).
2. *Check convergence.* If $\mathcal{N} = \emptyset$, then stop and declare infeasibility (if $\theta^{\mathrm{i}} = +\infty$) or report $(\boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}})$ as an optimal solution to problem (6.2).
3. *Select node.* Select a node $\tau = (\Xi_1, \dots, \Xi_K)$ from $\mathcal{N}$. Set $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\tau\}$.
4. *Process node.* Let $(\theta, \boldsymbol{x}, \boldsymbol{y})$ be an optimal solution to the scenario-based K-adaptability problem (6.11). If $\theta \ge \theta^{\mathrm{i}}$, then go to Step 2.
5. *Check feasibility.* Let $(\zeta, \boldsymbol{\xi}, \boldsymbol{z})$ be an optimal solution to the separation problem (6.13). If $\zeta \le 0$, then set $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}}) \leftarrow (\theta, \boldsymbol{x}, \boldsymbol{y})$ and go to Step 2.

6. *Branch.* Instantiate K new nodes $\tau_1, \ldots, \tau_K$ as follows: $\tau_k = (\Xi_1, \ldots, \Xi_k \cup \{\boldsymbol{\xi}\}, \ldots, \Xi_K)$ for each $k \in \mathcal{K}$. Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{\tau_1, \ldots, \tau_K\}$ and go to Step 3.

Our branch-and-bound algorithm can be interpreted as an uncertainty set partitioning scheme. For a solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ in Step 4, the sets

$$\Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_k) = \left\{ \boldsymbol{\xi} \in \Xi : \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta,\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\},\ \ k \in \mathcal{K},$$

describe the regions of the uncertainty set Ξ for which at least one of the candidate policies is feasible and results in an objective value smaller than or equal to θ. Step 5 of the algorithm attempts to identify a realization $\boldsymbol{\xi} \in \Xi \setminus \bigcup_{k \in \mathcal{K}} \Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_k)$ for which every candidate policy either is infeasible or results in an objective value that exceeds θ. If there is no such realization, then the solution $(\boldsymbol{x}, \boldsymbol{y})$ is feasible in the K-adaptability problem (6.2). Otherwise, Step 6 assigns the realization $\boldsymbol{\xi}$ to each scenario subset Ξ_k, $k \in \mathcal{K}$, in turn. Figure 6.3 illustrates our solution scheme.

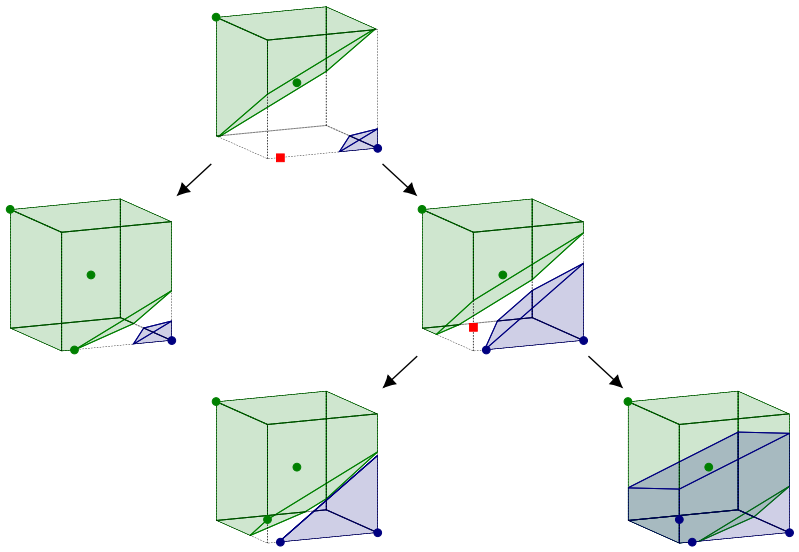


Figure 6.3: An illustrative example with $K = 2$ policies. Each cube represents the uncertainty set Ξ while the shaded regions represent $\Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_1)$ and $\Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_2)$. The green and blue dots represent elements of the sets Ξ_1 and Ξ_2, respectively, while the red squares represent the candidate realizations $\boldsymbol{\xi}$ identified in Step 5 of the algorithm.

6.3.2 *Convergence Analysis*

We now establish the correctness of our branch-and-bound scheme, as well as conditions for its asymptotic and finite convergence.

Theorem 6.1 (Correctness). *If the branch-and-bound scheme terminates, then it either returns an optimal solution to problem* (6.2) *or correctly identifies the latter as infeasible.*

Proof. We first show that if the problem instance is infeasible, then the algorithm terminates with the incumbent solution $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}}) = (+\infty, \emptyset, \emptyset)$. Indeed, the algorithm can only update the incumbent solution in Step 5 if the objective value of the separation problem is non-positive. By construction, this is only possible if the algorithm has determined a feasible solution.

We now show that for feasible problem instances, the algorithm terminates with an optimal solution $(\boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}})$ of problem (6.2). To this end, assume that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an optimal solution of problem (6.2) with objective value $\theta^\star$. Let $\mathcal{T}$ be the set of all nodes of the branch-and-bound tree for which $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is feasible in the corresponding scenario-based K-adaptability problem (6.11). Note that $\mathcal{T} \neq \emptyset$ since $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is feasible in the root node. Let $\mathcal{T}' \subseteq \mathcal{T}$ be the set of those nodes which have children in $\mathcal{T}$ and consider the set $\mathcal{T}'' = \mathcal{T} \setminus \mathcal{T}'$; by construction, we have $\mathcal{T}'' \neq \emptyset$. Consider an arbitrary node $\tau \in \mathcal{T}''$. By definition of $\mathcal{T}''$, our algorithm has not branched τ. Since τ has been selected in Step 3, this is only possible if either *(i)* τ has been fathomed in Step 4 or if *(ii)* τ has been fathomed in Step 5. In the former case, the solution $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$ must have been weakly dominated by the incumbent solution $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}})$, which therefore must be optimal as well. In the latter case, the incumbent solution must have been updated to $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$. □

We now show that our branch-and-bound scheme converges asymptotically to an optimal solution of the K-adaptability problem (6.2). Our result has two implications: *(i)* for infeasible problem instances, the algorithm always terminates after finitely many iterations, i.e., infeasibility is detected in finite time; *(ii)* for feasible problem instances, the algorithm eventually only inspects solutions in the neighborhood of optimal solutions.

Theorem 6.2 (Asymptotic Convergence)**.** *Every accumulation point $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the solutions to the scenario-based K-adaptability problem (6.11) in an infinite branch of the branch-and-bound tree gives rise to an optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the K-adaptability problem (6.2) with objective value $\hat{\theta}$.*

Proof. We denote by $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ the sequences of optimal solutions to the scenario-based K-adaptability problem in Step 4 and the separation problem in Step 5 of the algorithm, respectively, that correspond to the node sequence τ^ℓ, $\ell = 0, 1, \ldots$, of some infinite branch of the branch-and-bound tree. Since $\mathcal{X}$, $\mathcal{Y}$ and Ξ are compact, the Bolzano-Weierstrass theorem implies that $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ each have at least one accumulation point.

We first show that every accumulation point $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the sequence $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ corresponds to a *feasible* solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the K-adaptability problem (6.2) with objective value $\hat{\theta}$. By possibly going over to subsequences, we can without loss of generality assume that the two sequences $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ converge themselves to $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $(\hat{\zeta}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{z}})$, respectively. Assume now that $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ does *not* correspond to a feasible solution of the K-adaptability problem (6.2) with objective value $\hat{\theta}$. Then there is $\boldsymbol{\xi}^\star \in \Xi$ such that

$S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\xi}^\star) \geq \delta$ for some $\delta > 0$. By construction of the separation problem (6.13), this implies that

$$S(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell, \boldsymbol{\xi}^\ell) \;=\; \max_{\boldsymbol{\xi} \in \Xi} S(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell, \boldsymbol{\xi}) \;\geq\; S(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell, \boldsymbol{\xi}^\star) \;\geq\; \delta/2$$

for all ℓ sufficiently large. By taking limits and exploiting the continuity of S, we conclude that

$$S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}}) \;\geq\; S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\xi}^\star) \;\geq\; \delta/2.$$

Note, however, that $S(\theta^{\ell+1}, \boldsymbol{x}^{\ell+1}, \boldsymbol{y}^{\ell+1}, \boldsymbol{\xi}^\ell) \leq 0$ since $\boldsymbol{\xi}^\ell \in \Xi_k^{\ell+1}$ for some $k \in \mathcal{K}$. Since the sequence $(\theta^{\ell+1}, \boldsymbol{x}^{\ell+1}, \boldsymbol{y}^{\ell+1})$ also converges to $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $\boldsymbol{\xi}^\ell$ converges to $\hat{\boldsymbol{\xi}}$, we thus conclude that $S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}}) \leq 0$, which yields the desired contradiction.

We now show that every accumulation point $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the sequence $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ corresponds to an *optimal* solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the K-adaptability problem (6.2) with objective value $\hat{\theta}$. Assume to the contrary that $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is feasible but suboptimal. Then there is a feasible solution $(\theta', \boldsymbol{x}', \boldsymbol{y}')$ with $\theta' < \hat{\theta}$ that either *(i)* is used to update the incumbent solution after finitely many iterations, or *(ii)* constitutes the accumulation point of another infinite sequence $(\theta'^{,\ell}, \boldsymbol{x}'^{,\ell}, \boldsymbol{y}'^{,\ell})$. In the first case, the objective values θ^ℓ of the scenario-based K-adaptability problems will be arbitrarily close to $\hat{\theta}$ for ℓ sufficiently large, which implies that the corresponding nodes τ^ℓ will be fathomed in Step 4. Similarly, in the second case the objective values θ^ℓ and $\theta'^{,\ell}$ of the scenario-based K-adaptability problems will be arbitrarily close to $\hat{\theta}$ and θ', respectively, for ℓ sufficiently large. Since $\theta' < \hat{\theta}$, the algorithm will fathom the tree nodes corresponding to the sequence $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ in Step 4. The result now follows since both cases contradict the assumption that $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is an accumulation point. □

Theorem 6.2 guarantees that after sufficiently many iterations of the algorithm, our scheme generates feasible solutions that are close to an optimal solution of the K-adaptability problem (6.2). In general, our algorithm may not converge after finitely many iterations. In the following, we discuss a class of problem instances for which finite convergence is guaranteed.

Theorem 6.3 (Finite Convergence)**.** *The branch-and-bound scheme terminates after finitely many iterations, if $\mathcal{Y}$ has finite cardinality and only the objective function in problem (6.2) is uncertain.*

Proof. If only the objective function in the K-adaptability problem (6.2) is uncertain, then the corresponding semi-infinite disjunctive program (6.10) can be written as

$$\begin{array}{lll} \text{minimize} & \theta & \\ \text{subject to} & \theta \in \mathbb{R},\;\; \boldsymbol{x} \in \mathcal{X},\;\; \boldsymbol{y} \in \mathcal{Y}^K & \\ & \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h} & \forall k \in \mathcal{K} \\ & \bigvee\limits_{k \in \mathcal{K}} \left[\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \right] & \forall \boldsymbol{\xi} \in \Xi, \end{array}$$

see Observation 6.2. Thus, the scenario-based K-adaptability problem (6.11) becomes

$$\begin{array}{rcll} \mathcal{M}(\Xi_1,\ldots,\Xi_K) \;=\; & \text{minimize} & \theta & \\ & \text{subject to} & \theta \in \mathbb{R},\;\; \boldsymbol{x} \in \mathcal{X},\;\; \boldsymbol{y} \in \mathcal{Y}^K & \\ & & \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h} & \forall k \in \mathcal{K} \\ & & \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta & \forall \boldsymbol{\xi} \in \Xi_k,\; \forall k \in \mathcal{K}, \end{array}$$

and the separation problem (6.12) can be written as

$$\begin{aligned} \mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) \;&=\; \max_{\boldsymbol{\xi}\in\Xi} \min_{k\in\mathcal{K}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta \right\} \\ &=\; \boldsymbol{c}^\top \boldsymbol{x} - \theta + \max_{\boldsymbol{\xi}\in\Xi} \min_{k\in\mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \right\}. \end{aligned}$$

We now show that if $\mathcal{Y}$ has finite cardinality, then our branch-and-bound algorithm terminates after finitely many iterations. To this end, assume that this is not the case, and let τ^ℓ, $\ell = 0, 1, \ldots$ be some rooted branch of the tree with infinite length. We denote by $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ the corresponding sequences of optimal solutions to the master and the separation problem, respectively. Since $\mathcal{Y}$ has finite cardinality, we must have $\boldsymbol{y}^{\ell_1} = \boldsymbol{y}^{\ell_2}$ for some $\ell_1 < \ell_2$.

The solution $(\theta^{\ell_2}, \boldsymbol{x}^{\ell_2}, \boldsymbol{y}^{\ell_2})$ satisfies $\mathcal{S}(\theta^{\ell_2}, \boldsymbol{x}^{\ell_2}, \boldsymbol{y}^{\ell_2}) > 0$ since τ^ℓ, $\ell = 0, 1, \ldots$, is a branch of infinite length. Since $\boldsymbol{y}^{\ell_2} = \boldsymbol{y}^{\ell_1}$, we thus conclude that

$$\boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \max_{\boldsymbol{\xi}\in\Xi} \min_{k\in\mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k^{\ell_1} \right\} \;>\; 0.$$

Since $\boldsymbol{\xi}^{\ell_1}$ is optimal in the separation problem $\mathcal{S}(\theta^{\ell_1}, \boldsymbol{x}^{\ell_1}, \boldsymbol{y}^{\ell_1})$ and $S(\theta^{\ell_1}, \boldsymbol{x}^{\ell_1}, \boldsymbol{y}^{\ell_1}, \boldsymbol{\xi}^{\ell_1}) > 0$, we have

$$\boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \min_{k\in\mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi}^{\ell_1})^\top \boldsymbol{y}_k^{\ell_1} \right\} \;=\; \boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \max_{\boldsymbol{\xi}\in\Xi} \min_{k\in\mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k^{\ell_1} \right\} \;>\; 0.$$

However, since the node $\tau^{\ell_2} = (\Xi_1^{\ell_2}, \ldots, \Xi_K^{\ell_2})$ is a descendant of the node $\tau^{\ell_1} = (\Xi_1^{\ell_1}, \ldots, \Xi_K^{\ell_1})$, we must have $\boldsymbol{\xi}^{\ell_1} \in \Xi_k^{\ell_2}$ for some $k \in \mathcal{K}$. This, along with the fact that $(\theta^{\ell_2}, \boldsymbol{x}^{\ell_2}, \boldsymbol{y}^{\ell_2})$ is a feasible solution to the master problem $\mathcal{M}(\Xi_1^{\ell_2}, \ldots, \Xi_K^{\ell_2})$ and that $\boldsymbol{y}^{\ell_2} = \boldsymbol{y}^{\ell_1}$, implies that

$$\boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \min_{k\in\mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi}^{\ell_1})^\top \boldsymbol{y}_k^{\ell_1} \right\} \;\leq\; 0.$$

This yields the desired contradiction and proves the theorem. □

We note that the assumption of deterministic constraints is critical in the previous statement.

Example 6.2. *Consider the following instance of the K-adaptability problem (6.2):*

$$\inf_{y_1,y_2\in\{0,1\}} \sup_{\xi\in[0,1]} \inf_{k\in\{1,2\}} \{(\xi-1)(1-2y_k) \,:\, y_k \geq \xi\}$$

On this instance, our branch-and-bound algorithm generates a tree in which all branches have finite length, except (up to permutations) the sequence of nodes $\tau^\ell = (\Xi_1^\ell, \Xi_2^\ell)$, where $(\Xi_1^0, \Xi_2^0) = (\emptyset, \emptyset)$ and $(\Xi_1^\ell, \Xi_2^\ell) = \big(\{\xi^0 2^{-i} \,:\, i = 0, 1, \ldots, \ell-1\}, \{0\}\big)$, $\ell > 0$, for some $\xi^0 \in (0, 1]$. For the node τ^ℓ, $\ell > 1$, the optimal solution of the scenario-based K-adaptability problem (6.11) is $(\theta^\ell, y_1^\ell, y_2^\ell) = (1 - \xi^0 2^{-\ell+1}, 1, 0)$, while the optimal solution of the separation problem is $(\zeta^\ell, \xi^\ell) = (\xi^0 2^{-\ell}, \xi^0 2^{-\ell})$. Thus, our branch-and-bound algorithm does not terminate after finitely many iterations.

We note that every practical implementation of our branch-and-bound scheme will fathom nodes in Step 5 whenever the objective value of the separation problem (6.12) is sufficiently close to zero (within some ϵ-tolerance). This ensures that the algorithm terminates in finite time in practice. Indeed, in Example 6.2 the objective value of the separation problem is less than ϵ for all nodes τ^ℓ with $\ell \geq \log_2(\xi^0 \epsilon^{-1})$, and our branch-and-bound algorithm will fathom the corresponding path of the tree after $\mathcal{O}(\log \epsilon^{-1})$ iterations if we seek ϵ-precision solutions.

6.3.3 *Improvements to the Basic Algorithm*

The algorithm of Section 6.3.1 serves as a blueprint that can be extended in multiple ways. In the following, we discuss three enhancements that improve the numerical performance of our algorithm.

BREAKING SYMMETRY. For any feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ of the K-adaptability problem (6.2), every solution $(\boldsymbol{x}, \boldsymbol{y}')$, where $\boldsymbol{y}'$ is one of the $K!$ permutations of the second-stage policies $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K)$, is also feasible in (6.2) and attains the same objective value. This implies that our branch-and-bound tree is highly isomorphic since the scenario-based problems (6.11) and (6.13) are identical (up to a permutation of the policies) across many nodes. We can reduce this undesirable symmetry by modifying Step 6 of our branch-and-bound scheme as follows:

6′. *Branch.* Let $K' = 1$ if $\Xi_1 = \ldots = \Xi_K = \emptyset$ and let $K' = \min\Big\{K, 1 + \max_{k\in\mathcal{K}} \{k \,:\, \Xi_k \neq \emptyset\}\Big\}$ otherwise. Instantiate K' new nodes $\tau_k = (\Xi_1, \ldots, \Xi_k \cup \{\boldsymbol{\xi}\}, \ldots, \Xi_K)$, $k = 1, \ldots, K'$. Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{\tau_1, \ldots, \tau_{K'}\}$ and go to Step 3.

Despite generating only a subset of the nodes that our original algorithm constructs, the modification above always maintains at least one of the $K!$ solutions symmetric to every feasible solution.

INTEGRATION INTO MILP SOLVERS. Step 4 of our algorithm solves the scenario-based problem (6.11) from scratch in every node, despite the fact that two successive problems along any branch of the branch-and-bound tree differ only by the addition of a few constraints. We can leverage this commonality if we integrate our branch-and-bound algorithm into the solution scheme of the MILP solver used for problem (6.11). In doing so, we can also exploit the advanced facilities commonly present in the state-of-the-art solvers such as warm-starts and cutting planes, among others.

In order to integrate our branch-and-bound algorithm into the solution scheme of the MILP solver, we initialize the solver with the scenario-based problem (6.11) corresponding to the root node τ^0 of our algorithm, see Step 1. The solver then proceeds to solve this problem using its own branch-and-bound procedure. Whenever the solver encounters an *integral solution* $(\theta, \boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R} \times \mathcal{X} \times \mathcal{Y}^K$, we solve the associated separation problem (6.13). If $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) > 0$, then we execute Step 6 of our algorithm through a *branch callback*: we report the K new branches to the solver, which will discard the current solution. Otherwise, if $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) \leq 0$, then we do not create any new branches, and the solver will accept $(\theta, \boldsymbol{x}, \boldsymbol{y})$ as the new incumbent solution. This ensures that only those solutions which are feasible in problem (6.10) are accepted as incumbent solutions.

Whenever the solver encounters a *fractional solution*, it will by default branch on an integer variable that is fractional in the current solution. However, if $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) > 0$, it is possible to override this strategy and instead execute Step 6 of our algorithm. In such cases, a heuristic rule can be used to decide whether to branch on integer variables or to branch as in Step 6. In our computational experience, a simple rule that alternates between the default branching rule of the solver and the one defined by Step 6 appears to perform well in practice.

6.3.4 *Modification as a Heuristic Algorithm*

Whenever the number of policies K is large, the solution of the scenario-based K-adaptability problem (6.11) can be time consuming. In such cases, only a limited number of nodes will be explored by the algorithm in a given amount of computation time, and the quality of the final incumbent solution may be poor. As a remedy, we can reduce the size and complexity of the scenario-based K-adaptability problem (6.11) by fixing some of its second-stage policies. In doing so, we obtain a heuristic variant of our algorithm that can scale to large values of K.

In our computational experience, a simple heuristic that sequentially solves the 1-, 2-, ..., K-adaptability problems by fixing in each K-adaptability problem all but one of the second-stage policies, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{K-1}$, to their corresponding values in the $(K-1)$-adaptability problem, performs well in practice. This heuristic is motivated by two observations. First, the resulting scenario-based K-adaptability problems (6.11) have the same size and complexity as the corresponding scenario-based 1-adaptability problems. Second, in our experiments on instances with uncertain objective coefficients $\boldsymbol{d}$, we often

found that some optimal second-stage policies of the $(K-1)$-adaptability problem also appear in the optimal solution of the K-adaptability problem. In fact, it can be shown that this heuristic can obtain K-adaptable solutions that improve upon 1-adaptable solutions only if the objective coefficients $\boldsymbol{d}$ are affected by uncertainty.

6.4 NUMERICAL RESULTS

We now analyze the computational performance of our branch-and-bound scheme in a variety of problem instances from the literature. We consider a shortest path problem with uncertain arc weights (Section 6.4.1), a capital budgeting problem with uncertain cash flows (Section 6.4.2), a variant of the capital budgeting problem with the additional option to take loans (Section 6.4.3), a project management problem with uncertain task durations (Section 6.4.4), and a vehicle routing problem with uncertain travel times (Section 6.4.5). Of these, the first two problems involve only binary decisions, and they can therefore also be solved with the approach described in [145]. In these cases, we show that our solution scheme is highly competitive, and it frequently outperforms the approach of [145]. In contrast, the third and fourth problems also involve continuous decisions, and there is no existing solution approach for their associated K-adaptability problems. However, the project management problem from Section 6.4.4 involves *only* continuous second-stage decisions, and therefore the corresponding two-stage robust optimization problem (6.1) can also be approximated using affine decision rules [38], which represent the most popular approach for such problems. In this case, we elucidate the benefits of K-adaptable constant and affine decisions over standard affine decision rules. Finally, the first and last problems involve only binary second-stage decisions and deterministic constraints, and they can therefore also be addressed with the heuristic approach described in [69]. In these cases, we show that the heuristic variant of our algorithm often outperforms the latter approach in terms of solution quality.

For each problem category, we investigate the tradeoffs between computational effort and improvement in objective value of the K-adaptability problem for increasing values of K. We demonstrate that *(i)* the K-adaptability problem can provide significant improvements over static robust optimization (which corresponds to the case $K = 1$), and that *(ii)* our solution scheme can quickly determine feasible solutions of high quality.

We implemented our branch-and-bound algorithm in C++ using the C callable library of CPLEX 12.7 [154]. We used a constraint feasibility tolerance of $\epsilon = 10^{-4}$ to accept any incumbent solutions, whereas all other solver options were kept at their default values. The experiments were conducted on a single core of an Intel Xeon 2.8GHz computer with 16GB RAM.

6.4.1 *Shortest Paths*

We consider the shortest path problem from [145]. Let $G = (V, A)$ be a directed graph with nodes $V = \{1, \ldots, N\}$, arcs $A \subseteq V \times V$ and arc weights $d_{ij}(\boldsymbol{\xi}) = (1 + \xi_{ij}/2)d^0_{ij}$, $(i, j) \in A$. Here, $d^0_{ij} \in \mathbb{R}_+$ represents the nominal weight of the arc $(i, j) \in A$ and ξ_{ij} denotes the uncertain deviation from the nominal weight. The realizations of the uncertain vector $\boldsymbol{\xi}$ are known to belong to the set

$$\Xi = \left\{ \boldsymbol{\xi} \in [0,1]^{|A|} : \sum_{(i,j)\in A} \xi_{ij} \leq \Gamma \right\},$$

which stipulates that at most Γ arc weights may maximally deviate from their nominal values.

Let $s \in V$ and $t \in V$, $s \neq t$, denote the source and terminal nodes of G, respectively. The decision-maker aims to choose K paths from s to t here-and-now, i.e., before observing the actual arc weights, such that the worst-case weight of the shortest among the chosen paths is minimized. This problem can be formulated as an instance of the K-adaptability problem (6.2):

$$\inf_{\boldsymbol{y}\in\mathcal{Y}^K} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k$$

Here, $\mathcal{Y}$ denotes the set of all $s - t$ paths in G; that is,

$$\mathcal{Y} = \left\{ \boldsymbol{y} \in \{0,1\}^{|A|} : \sum_{(j,l)\in A} y_{jl} - \sum_{(i,j)\in A} y_{ij} \geq \mathbb{I}[j = s] - \mathbb{I}[j = t] \;\; \forall\, j \in V \right\}.$$

Note that this problem only contains second-stage decisions and as such, the corresponding two-stage robust optimization problem (6.1) may be of limited interest in practice. Nevertheless, the K-adaptability problem (6.2) has important applications in logistics and disaster relief [145].

For each graph size $N \in \{20, 25, \ldots, 50\}$, we randomly generate 100 problem instances as follows. We assign the coordinates $(u_i, v_i) \in \mathbb{R}^2$ to each node $i \in V$ uniformly at random from the square $[0, 10]^2$. The nominal weight of the arc $(i, j) \in A$ is defined to be the Euclidean distance between the nodes i and j; that is, $d^0_{ij} = \sqrt{(u_i - u_j)^2 + (v_i - v_j)^2}$. The source node s and the terminal node t are defined to be the nodes with the maximum Euclidean distance between them. The arc set A is obtained by removing from the set of all pairwise links the $\lfloor 0.7(N^2 - N) \rfloor$ connections with the largest nominal weights. We set the uncertainty budget to $\Gamma = 3$. Further details on the parameter settings can be found in [145].

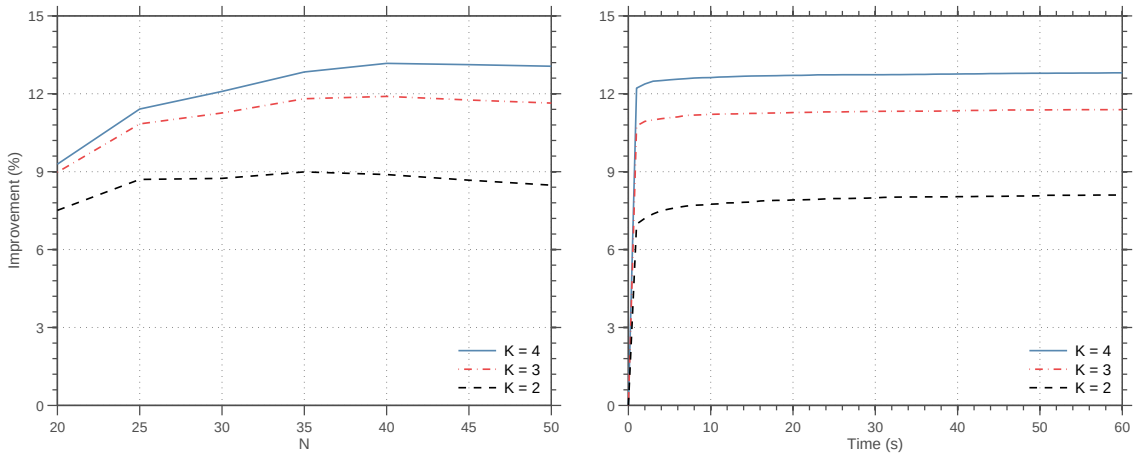
Table 6.2 summarizes the numerical performance of our branch-and-bound scheme for $K \in \{2, 3, 4\}$. Table 6.2 indicates that our scheme is able to reliably compute optimal

solutions for small values of N and K, while the average optimality gap for large values of N and K is less than 9%. The numerical performance is strongly affected by the value of K; very few of the 4-adaptable instances are solved to optimality within the time limit. This decrease in tractability is partly explained in Figure 6.4, which shows the improvement in objective value of the K-adaptability problem over the static problem (where $K = 1$). Figure 6.4a shows that the computed 4-adaptable solutions are typically of high quality since they improve upon the static solutions by as much as 13% for large values of N. Moreover, Figure 6.4b shows that these solutions are obtained within 1 minute (on average), even for the largest instances. This indicates that the gaps in Table 6.2 are likely to be very conservative since the majority of computation time is spent on obtaining a certificate of optimality for these solutions.

Table 6.2: Results for the shortest path problem. For each value of K, the "Opt (#)" column reports the number of instances (out of 100) which were solved to optimality, while the "Time (s)" column reports the average time to solve these instances to optimality. For those instances which could not be solved to optimality within the time limit of 7,200s, the average gap $|(\text{ub} - \text{lb})/\text{ub}| \times 100\%$ between the global lower bound (lb) and global upper bound (ub) of the branch-and-bound tree is reported in the "Gap (%)" column.

	$K = 2$			$K = 3$			$K = 4$		
N	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)
20	99	6	1.23	97	408	2.51	70	539	1.74
25	91	222	4.14	64	847	2.91	33	885	2.89
30	64	744	4.40	31	1,237	4.10	16	827	4.27
35	37	1,083	5.36	14	1,020	5.01	10	896	5.23
40	10	808	6.28	6	1,670	6.43	2	39	6.10
45	9	1,152	7.70	1	16	7.06	1	15	6.61
50	2	3,307	8.55	1	2,308	7.90	0	–	7.10

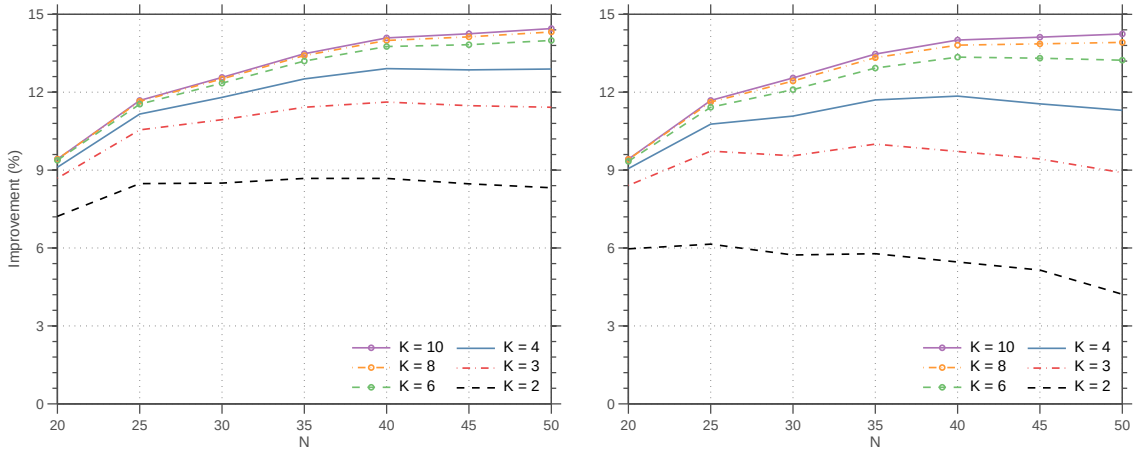
Figure 6.5 illustrates the quality of the solutions obtained using the heuristic variant of our algorithm, described in Section 6.3.4, and contrasts it with the quality of the solutions obtained using the heuristic algorithm described in [69]. Figure 6.5 shows that, after just one minute of computation time, the 2-, 3- and 4-adaptable solutions obtained using our heuristic algorithm are within 0.3% of known optimal solutions and about 2% better than those obtained using the heuristic algorithm described in [69], on average. The differences in the qualities of the 6-, 8- and 10-adaptable solutions are smaller. The figure also shows that the marginal gain in objective value decreases rapidly as we increase the number of policies K. Indeed, while the 2-adaptable solutions are about 8.3% better than the 1-adaptable (i.e., static) solutions, the 10-adaptable solutions are only about 0.1% better than the 8-adaptable solutions. This may be explained by the possibility that the



(a) (b)

Figure 6.4: Results for the shortest path problem using the exact algorithm. The graphs show the average improvement $|(\theta_1 - \theta_K)/\theta_1| \times 100\%$ of the objective value of the K-adaptability problem (θ_K) over the static problem (θ_1). The left graph shows the improvement after 2 hours (for increasing N), while the right graph shows the time profile of the improvement of the incumbent solution in the first 60 seconds (for $N = 50$).

objective values of the corresponding K-adaptable solutions are very close to the optimal value of the two-stage robust optimization problem (6.1).



(a) (b)

Figure 6.5: Results for the shortest path problem using the heuristic algorithm. The graphs show the average improvement after 1 minute obtained using the heuristic variant of our algorithm (left) and the heuristic algorithm described in [69] (right).

6.4.2 *Capital Budgeting*

We consider the capital budgeting problem from [145], where a company wishes to invest in a subset of N projects. Each project i has an uncertain cost $c_i(\boldsymbol{\xi})$ and an uncertain profit $r_i(\boldsymbol{\xi})$ that are governed by factor models of the form

$$c_i(\boldsymbol{\xi}) = \left(1 + \boldsymbol{\Phi}_i^\top \boldsymbol{\xi}/2\right) c_i^0 \quad \text{and} \quad r_i(\boldsymbol{\xi}) = \left(1 + \boldsymbol{\Psi}_i^\top \boldsymbol{\xi}/2\right) r_i^0 \quad \text{for } i = 1, \ldots, N.$$

In these models, c_i^0 and r_i^0 represent the nominal cost and the nominal profit of project i, respectively, while $\boldsymbol{\Phi}_i^\top$ and $\boldsymbol{\Psi}_i^\top$ represent the i^{th} row vectors of the factor loading matrices $\boldsymbol{\Phi}, \boldsymbol{\Psi} \in \mathbb{R}^{N\times 4}$. The realizations of the uncertain vector of risk factors $\boldsymbol{\xi}$ belong to the uncertainty set $\Xi = [-1, 1]^4$.

The company can invest in a project either before or after observing the risk factors $\boldsymbol{\xi}$. In the latter case, the company generates only a fraction κ of the profit (reflecting a penalty for postponement) but incurs the same cost as in the case of an early investment. Given an investment budget B, the capital budgeting problem can then be formulated as the following instance of the two-stage robust optimization problem (6.1):

$$\sup_{\boldsymbol{x}\in\mathcal{X}} \inf_{\boldsymbol{\xi}\in\Xi} \sup_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \boldsymbol{r}(\boldsymbol{\xi})^\top (\boldsymbol{x} + \kappa \boldsymbol{y}) \;:\; \boldsymbol{c}(\boldsymbol{\xi})^\top (\boldsymbol{x} + \boldsymbol{y}) \leq B,\; \boldsymbol{x} + \boldsymbol{y} \leq \mathbf{e} \right\},$$

where $\mathcal{X} = \mathcal{Y} = \{0, 1\}^N$.

For our numerical experiments, we randomly generate 100 instances for each problem size $N \in \{5, 10, \ldots, 30\}$ as follows. The nominal costs $\boldsymbol{c}^0$ are chosen uniformly at random from the hyperrectangle $[0, 10]^N$. We then set $\boldsymbol{r}^0 = \boldsymbol{c}^0/5$, $B = \mathbf{e}^\top \boldsymbol{c}^0/2$ and $\kappa = 0.8$. The rows of the factor loading matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are sampled uniformly from the unit simplex in $\mathbb{R}^4$; that is, the i^{th} row vector is sampled from $[0, 1]^4$ such that $\boldsymbol{\Phi}_i^\top \mathbf{e} = \boldsymbol{\Psi}_i^\top \mathbf{e} = 1$ is satisfied for all $i = 1, \ldots, N$.

Table 6.3 summarizes the numerical performance of our branch-and-bound scheme for $K \in \{2, 3, 4\}$. Table 6.3 demonstrates that our branch-and-bound scheme performs very well for this problem class since the majority of instances is solved to optimality for $K \in \{2, 3\}$. Moreover, the optimality gaps for the unsolved instances are less than 4% for $K \in \{2, 3\}$ and less than 9% for $K = 4$ on average. Additionally, Figure 6.6 shows that even for the largest instances, high-quality incumbent solutions which significantly improve ($\approx$100%) upon the static robust solutions are obtained within 1 minute of computation time. Our results compare favorably with those of [145] as well as those of the partition-and-bound approach for the corresponding two-stage robust optimization problem presented in [44].

6.4.3 *Capital Budgeting with Loans*

We consider a generalization of the capital budgeting problem from Section 6.4.2 where the company can increase its investment budget by purchasing a loan from the bank

Table 6.3: Results for the capital budgeting problem. The columns have the same interpretation as in Table 6.2.

	$K = 2$			$K = 3$			$K = 4$		
N	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)
5	100	1	–	100	1	–	100	3	–
10	100	1	–	100	16	–	100	149	–
15	100	10	–	99	566	0.33	69	2,245	1.42
20	100	419	–	34	2,787	1.65	5	3,710	4.02
25	29	2,238	1.12	4	2,281	2.63	0	–	6.22
30	1	188	3.01	1	6,687	3.35	0	–	8.27

at a unit cost of $\lambda > 0$ before the risk factors $\boldsymbol{\xi}$ are observed as well as purchasing a loan at a unit cost of $\mu\lambda$ (with $\mu > 1$) after the observation occurs. If the company does not purchase any loan, then the problem reduces to the one described in Section 6.4.2. Therefore, we expect the worst-case profits to be at least as large as in that setting. The generalized capital budgeting problem can be formulated as the following instance of problem (6.1):

$$\sup_{(x_0,\boldsymbol{x})\in\mathcal{X}} \inf_{\boldsymbol{\xi}\in\Xi} \sup_{(y_0,\boldsymbol{y})\in\mathcal{Y}} \left\{ \boldsymbol{r}(\boldsymbol{\xi})^\top(\boldsymbol{x}+\kappa\boldsymbol{y}) - \lambda(x_0+\mu y_0) \;:\; \begin{bmatrix} \boldsymbol{x}+\boldsymbol{y} \leq \mathbf{e} \\ \boldsymbol{c}(\boldsymbol{\xi})^\top\boldsymbol{x} \leq B + x_0 \\ \boldsymbol{c}(\boldsymbol{\xi})^\top(\boldsymbol{x}+\boldsymbol{y}) \leq B + x_0 + y_0 \end{bmatrix} \right\}$$

Here, $\mathcal{X} = \mathcal{Y} = \mathbb{R}_+ \times \{0,1\}^N$. The constraint $\boldsymbol{c}(\boldsymbol{\xi})^\top\boldsymbol{x} \leq B + x_0$ ensures that the first-stage expenditures $\boldsymbol{c}(\boldsymbol{\xi})^\top\boldsymbol{x}$ are fully covered by the budget B as well as the loan x_0 taken here-and-now.

We consider problems with $N \in \{5, 10, \ldots, 30\}$ projects. For each value of N, we solve the same 100 instances from Section 6.4.2 with $\lambda = 0.12$ and $\mu = 1.2$. Table 6.4 shows the computational performance of our branch-and-bound scheme for $K \in \{2, 3, 4\}$. As in the case of the problems discussed so far, the numerical tractability of our algorithm decreases as the value of K increases. However, a comparison of Tables 6.3 and 6.4 suggests that the numerical tractability is not significantly affected by the presence of the additional continuous variables x_0 and y_0. Indeed, the majority of instances for $K = 2$ are solved to optimality and the average gap across all unsolved instances is less than 5% for $K = 3$ and less than 9% for $K = 4$. Figure 6.7 shows that the 4-adaptable solutions improve upon the static solutions by as much as 115% in the largest instances. Although not shown in the figure, a comparison of the objective values of the final incumbent solutions with those of the capital budgeting problem without loans (Section 6.4.2) reveals that for $N \geq 15$, the option to purchase loans has no effect on the worst-case profit of the static solution and results in less than 1% improvement in the worst-case profit of the

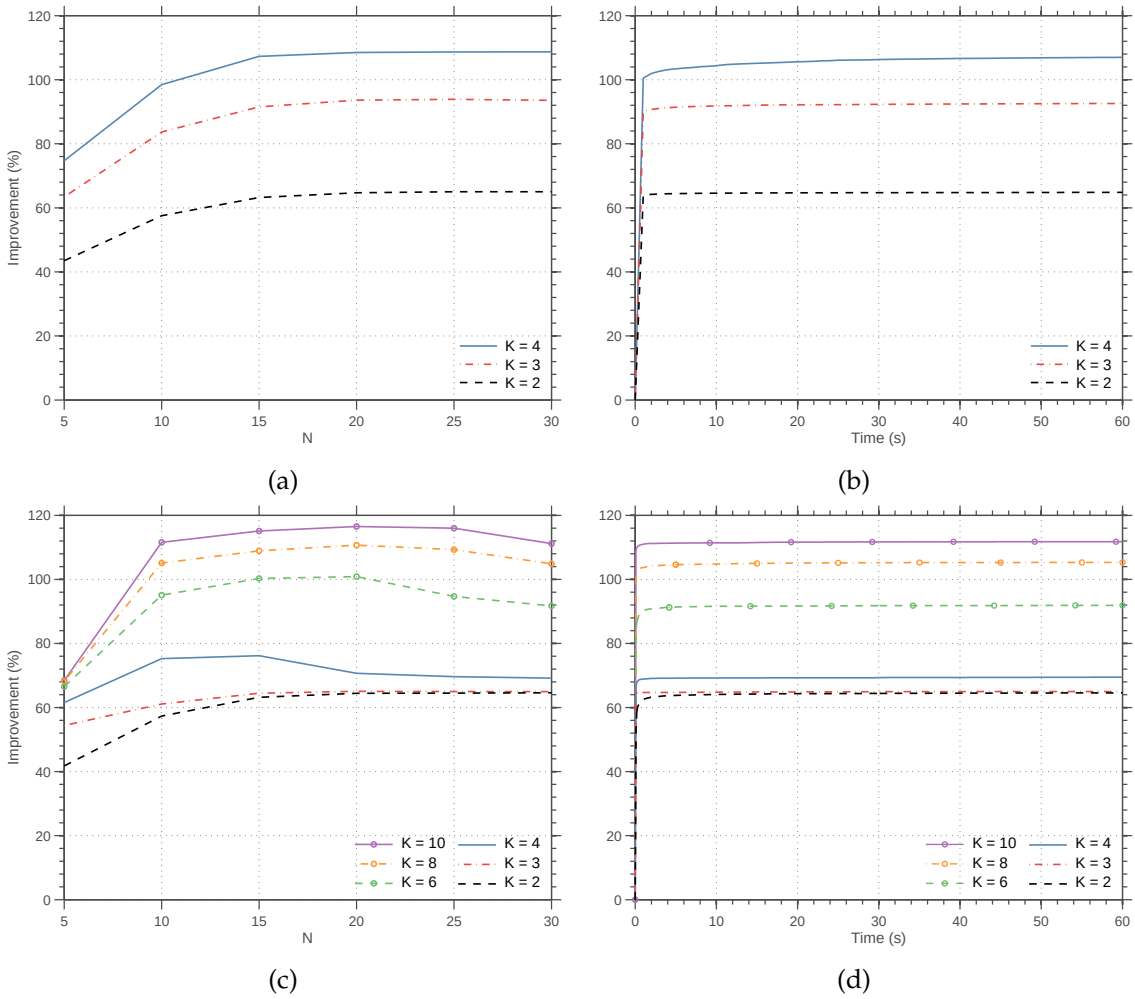


Figure 6.6: Results for the capital budgeting problem. The top (bottom) graphs are for the exact (heuristic) algorithm under a time limit of 2 hours (1 minute). The left graphs show the average improvement at the time limit (for increasing N), while the right graphs show the time profile of the improvement of the incumbent solution in the first 60 seconds (for $N = 30$).

2-adaptable solution. Indeed, the option to purchase loans results in significantly better worst-case profits only if $K \geq 3$. The average relative gain in objective value is 4.3% for $K = 3$ and 5.9% for $K = 4$.

6.4.4 *Project Management*

We define a project as a directed acyclic graph $G = (V, A)$ whose nodes $V = \{1, \ldots, N\}$ represent the tasks (e.g., 'build foundations' or 'develop prototype') and whose arcs $A \subseteq V \times V$ denote the temporal precedences, i.e., $(i, j) \in A$ implies that task j cannot

Table 6.4: Results for the capital budgeting problem with loans. The columns have the same interpretation as in Table 6.2.

	$K = 2$			$K = 3$			$K = 4$		
N	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)
5	100	1	–	100	9	–	98	80	3.14
10	100	3	–	100	78	–	98	938	1.92
15	100	62	–	96	1,265	0.91	23	3,989	2.23
20	85	1,680	0.80	20	3,941	1.71	0	–	4.94
25	12	3,363	2.29	1	2,693	3.34	0	–	6.88
30	1	424	3.78	0	–	4.73	0	–	8.17

be started before task i has been completed. We assume that each task $i \in V$ has an uncertain duration $d_i(\boldsymbol{\xi})$ that depends on the realization of an uncertain parameter vector $\boldsymbol{\xi} \in \Xi$. Without loss of generality, we stipulate that the project graph G has the unique sink $N \in V$, and that the last task N has a duration of zero. This can always be achieved by introducing dummy nodes and/or arcs.

In the following, we want to calculate the worst-case makespan of the project, i.e, the smallest amount of time that is required to complete the project under the worst realization of the parameter vector $\boldsymbol{\xi} \in \Xi$. This problem can be cast as the following instance of problem (6.1):

$$\sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{y}\in\mathcal{Y}} \left\{ y_N \,:\, y_j - y_i \geq d_i(\boldsymbol{\xi}) \;\; \forall (i,j) \in A \right\}$$

Here $\mathcal{Y} = \mathbb{R}^N_+$, and y_i denotes the start time of task i, $i = 1, \ldots, N$. This problem is known to be NP-hard [268, Theorem 2.1], and we will employ affine decision rules as well as K-adaptable constant and affine decisions to approximate the optimal value of this problem. Note that the problem does not contain any first-stage decisions, but such decisions could be readily included, for example, to allow for resource allocations that affect the task durations.

For our numerical experiments, we consider the instance class presented in [268, Example 2.2]. To this end, we set $N = 3m + 1$ and $A = \{(3l+1, 3l+p), (3l+p, 3l+4) \,:\, l = 0, \ldots, m \text{ and } p = 2, 3\}$, $d_{3l+2}(\boldsymbol{\xi}) = \xi_{l+1}$ and $d_{3l+3}(\boldsymbol{\xi}) = 1 - \xi_{l+1}$, $l = 0, \ldots, m-1$, as well as $d_{3l+1}(\boldsymbol{\xi}) = 0$, $l = 0, \ldots, m$. Figure 6.8 illustrates the project network corresponding to $m = 4$. Similar to [268], we consider the uncertainty set $\Xi = \{\boldsymbol{\xi} \in \mathbb{R}^m_+ \,:\, \|\boldsymbol{\xi} - \mathbf{e}/2\|_1 \leq 1/2\}$.

We consider project networks of size $N \in \{3m + 1 \,:\, m = 3, 4, \ldots, 8\}$. One can show that for each network size N, the optimal value of the corresponding static robust optimization problem as well as the affine decision rule problem is m, see [268, Example 2.2]. For

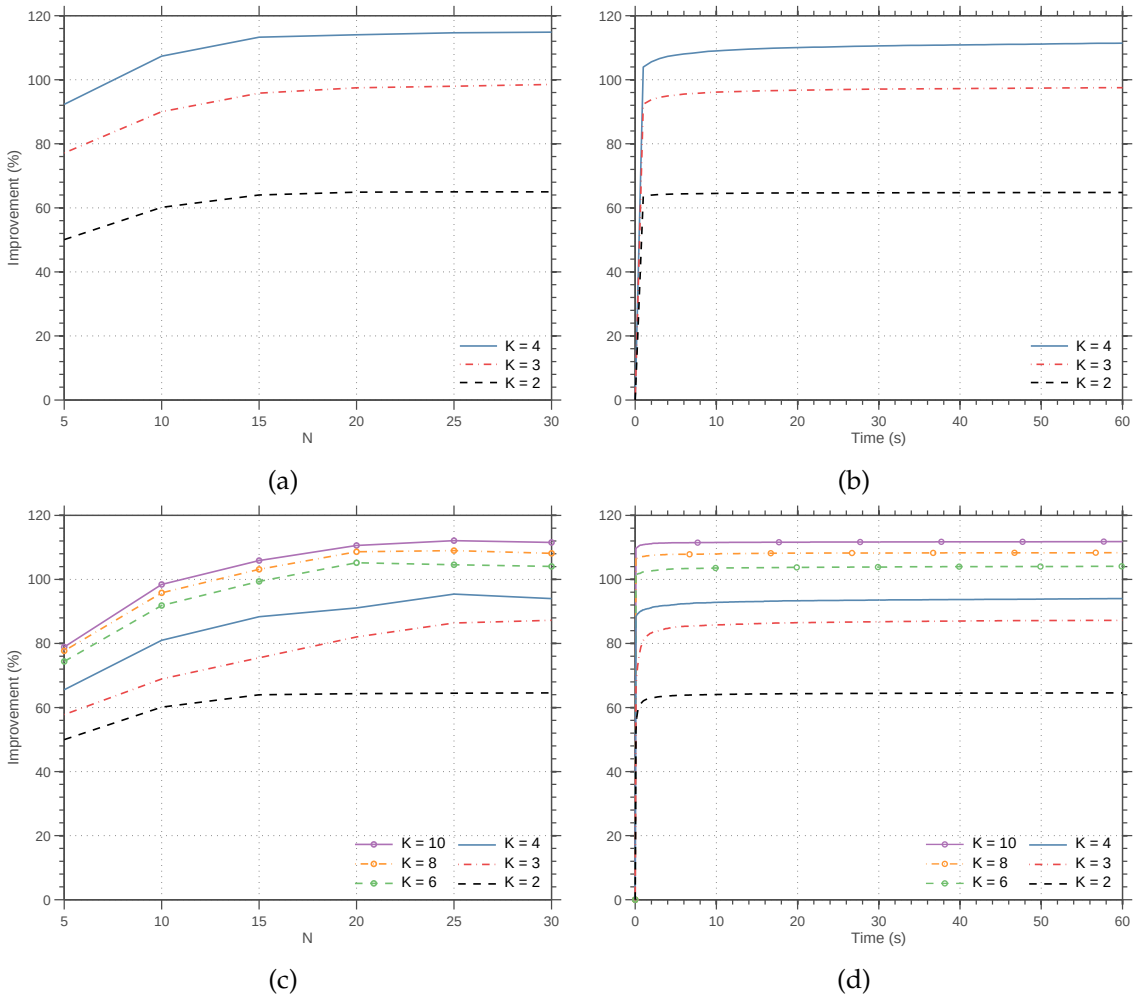


Figure 6.7: Results for the capital budgeting problem with loans. The graphs have the same interpretation as in Figure 6.6.

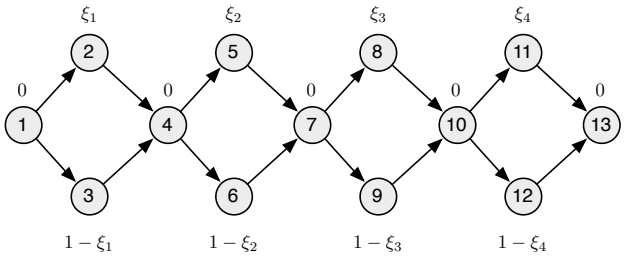


Figure 6.8: Project network with $N = 3m + 1$ nodes for $m = 4$.

$K \in \{2, 3, 4\}$, Figure 6.9 summarizes the computational performance of the branch-and-bound scheme and the improvement in objective value of the resulting piecewise constant and piecewise affine decision rules with K pieces over the corresponding 1-adaptable solutions. Figures 6.9a and 6.9b show that using only two pieces, piecewise constant

decision rules can improve upon the affine approximation by more than 12%, while a piecewise affine decision rule can improve by more than 15%. Figures 6.9c and 6.9d show that piecewise constant decision rules require smaller computation times than piecewise affine decision rules. This is not surprising since piecewise constant decision rules are parameterized by $\mathcal{O}(KN)$ variables, whereas piecewise affine decision rules are parameterized by $\mathcal{O}(KN^2)$ variables.

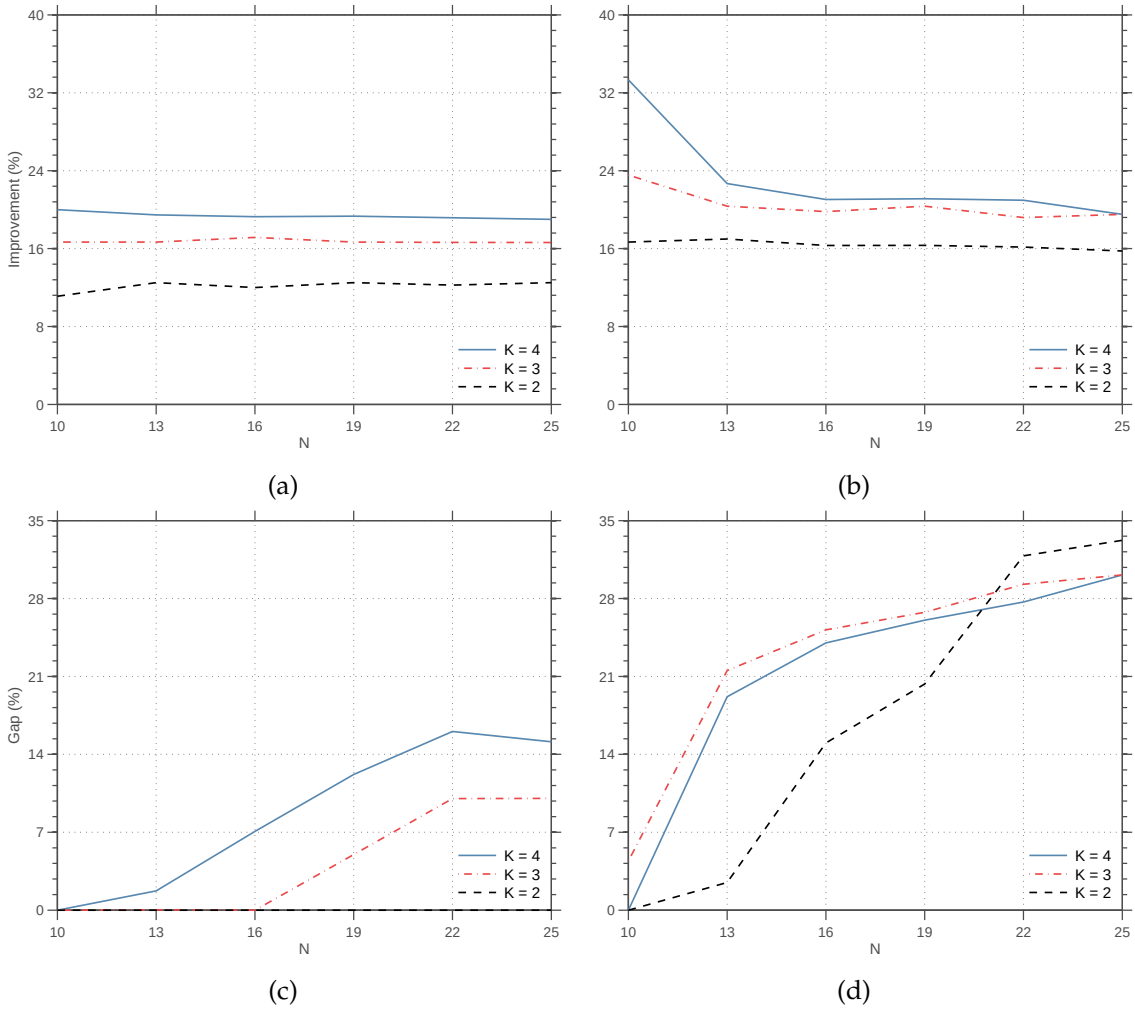


Figure 6.9: Results for the project management problem. The left (right) graphs show results for the piecewise constant (affine) K-adaptability problems for increasing values of N. Graphs (a) and (b) depict improvements in objective value, while graphs (c) and (d) show optimality gaps after 2 hours. The y-axes in graphs (a) and (b) have the same interpretation as those in Figure 6.4 while the y-axes in graphs (c) and (d) have the same interpretation as the column "Gap (%)" in Table 6.2.

6.4.5 *Vehicle Routing*

We consider the classical capacitated vehicle routing problem [137, 138, 256] defined on a complete, undirected graph $G = (V, E)$ with nodes $V = \{0, 1, \ldots, N\}$ and edges $E = \{(i, j) \in V \times V : i < j\}$. Node 0 represents the unique depot, while each node $i \in V_C = \{1, \ldots, N\}$ corresponds to a customer with demand $d_i \in \mathbb{R}_+$. The depot is equipped with M homogeneous vehicles; each vehicle has capacity C and it incurs an uncertain travel time $t_{ij}(\boldsymbol{\xi}) = (1 + \xi_{ij}/2)t^0_{ij}$ when it traverses the edge $(i, j) \in E$. Here, $t^0_{ij} \in \mathbb{R}_+$ represents the nominal travel time along the edge $(i, j) \in E$, while ξ_{ij} denotes the uncertain deviation from the nominal value. Similar to the shortest path problem from Section 6.4.1, the realizations of the uncertain vector $\boldsymbol{\xi}$ are known to belong to the set

$$\Xi = \left\{ \boldsymbol{\xi} \in [0,1]^{|E|} : \sum_{(i,j)\in E} \xi_{ij} \leq \Gamma \right\},$$

which stipulates that at most Γ travel times may maximally deviate from their nominal values.

A *route plan* $(R_1, \ldots, R_M)$ corresponds to a partition of the customer set V_C into M vehicle routes, $R_m = (R_{m,1}, \ldots, R_{m,N_m})$, where $R_{m,l}$ represents the l^{th} customer and N_m the number of customers served by the m^{th} vehicle. This route plan is feasible if the total demand served on each route is less than the vehicle capacity; that is, if $\sum_{l=1}^{N_m} d_{R_{m,l}} \leq C$ is satisfied for all $m \in \{1, \ldots, M\}$. The total travel time of a feasible route plan under the uncertainty realization $\boldsymbol{\xi}$ is given by $\sum_{m=1}^{M} \sum_{l=0}^{N_m} t_{R_{m,l}R_{m,l+1}}(\boldsymbol{\xi})$, where we define $R_{m,0} = R_{m,N_m+1} = 0$; that is, each vehicle starts and ends at the depot. The decision-maker aims to choose K route plans here-and-now, i.e., before observing the actual travel times, such that the worst-case total travel time of the shortest among the chosen route plans is minimized. This problem can be formulated as an instance of the K-adaptability problem (6.2):

$$\inf_{\boldsymbol{y}\in\mathcal{Y}^K} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \boldsymbol{t}(\boldsymbol{\xi})^\top \boldsymbol{y}_k$$

Here, $\mathcal{Y}$ denotes the set of all feasible route plans in G; that is,

$$\mathcal{Y} = \left\{ \boldsymbol{y} \in \mathbb{Z}_+^{|E|} : \begin{array}{l} 0 \leq y_{ij} \leq 1 \quad \forall (i,j) \in E : i, j \in V_C, \\ \sum_{j\in V_C} y_{0j} = 2M, \\ \sum_{j\in V:(i,j)\in E} y_{ij} = 2 \quad \forall i \in V_C, \\ \sum_{(i,j)\in E: i,j\in U} y_{ij} \leq |U| - \left\lceil \frac{1}{D} \sum_{i\in U} d_i \right\rceil \quad \forall U \subseteq V_C \end{array} \right\}.$$

Similar to the shortest path problem, the K-adaptability formulation of the vehicle routing problem only contains second-stage decisions, and as such, the corresponding two-stage

robust optimization problem (6.1) is of limited interest in practice. However, the K-adaptability problem (6.2) has important applications in logistics enterprises, where the time available between observing the travel times in a road network and determining the route plan is limited, or because the drivers must be trained to a small set of route plans that are to be executed daily over the course of a year.

We note that the set $\mathcal{Y}$ represents the so-called *two-index vehicle flow* formulation of the vehicle routing problem, in which the first equation ensures that M vehicles are used; the second set of equations ensures that each customer is visited by exactly one vehicle; while the third set of inequalities ensure that there are no *subtours* disconnected from the depot and that all vehicle capacities are respected. This formulation is known to be extremely challenging to solve because it consists of an exponential number of inequalities. For $K > 1$, the corresponding K-adaptability problem is naturally even more challenging and it is practically intractable to solve it using the approach described in [145]. In contrast, the heuristic variant of our algorithm described in Section 6.3.4 as well as the heuristic approach of [69] only require the solution of vehicle routing subproblems that are of similar complexity as the associated 1-adaptability problems. Therefore, in the following, we only present results using these algorithms. In both cases, we solved all vehicle routing subproblems using the branch-and-cut algorithm described in [193].

For our numerical experiments, we consider all 49 instances from [193] with $N \leq 50$, which are commonly used to benchmark vehicle routing algorithms. We set an overall time limit of 2 hours. For the heuristic variant of our algorithm, we further set a time limit of 10 minutes per vehicle routing subproblem. We note that the heuristic of [69] requires the successful termination of an expensive preprocessing step to determine good K-adaptable solutions. Therefore, to prevent bias in favor of our algorithm, Figure 6.10 compares the two algorithms only across the 39 instances for which this step terminated successfully. The figure shows that when the number of policies is small, the K-adaptable solutions obtained using our algorithm are about 1% better than those obtained using the heuristic algorithm of [69]. Moreover, the differences in their objective values are relatively higher for larger instances.

6.5 SUMMARY

In contrast to single-stage robust optimization problems, which are typically solved via monolithic reformulations, there is growing evidence that two-stage and multi-stage robust optimization problems are best solved algorithmically [44, 46, 47, 218, 275]. Our findings in this chapter appear to confirm this observation, as our proposed branch-and-bound algorithm compares favorably with the reformulations proposed in [145]. In terms of modeling flexibility, our algorithm can accommodate mixed continuous and discrete decisions in both stages, can incorporate discrete uncertainty, and allows us to model flexible piecewise affine decision rules. At the same time, our numerical results indicate that the algorithmic approach is highly competitive in terms of computational

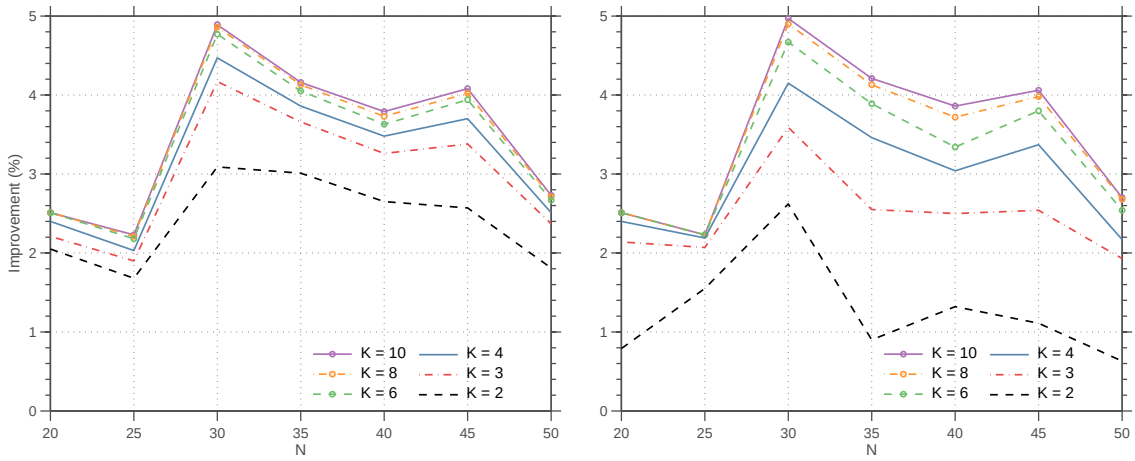


Figure 6.10: Results for the vehicle routing problem. The graphs show the average improvement after 2 hours obtained using the heuristic variant of our algorithm (left) and the heuristic algorithm in [69] (right).

performance as well. From a practical viewpoint, a notable feature of our algorithm is that it admits a lightweight implementation by integrating it into the branch-and-bound schemes of commercial solvers via branch callbacks, while allowing easy modification as a heuristic for large-scale instances.

6.6 APPENDIX: NOMENCLATURE

$\boldsymbol{x}$	First-stage (or here-and-now) decisions
$\boldsymbol{y}$	Second-stage (or wait-and-see) decisions
$\boldsymbol{\xi}$	Uncertain parameters
N_1	Number of first-stage decisions
N_2	Number of second-stage decisions
N_p	Number of uncertain parameters
$\mathcal{X}$	Domain of first-stage decisions
$\mathcal{Y}$	Domain of second-stage decisions
Ξ	Uncertainty set
$\boldsymbol{c}$	Objective coefficients of first-stage decisions
$\boldsymbol{d}$	Objective coefficients of second-stage decisions
$\boldsymbol{T}$	Technology matrix (constraint matrix of first-stage decisions)
$\boldsymbol{W}$	Recourse matrix (constraint matrix of second-stage decisions)
$\boldsymbol{h}$	Right-hand side vector (of problem constraints)
L	Number of constraints affected by uncertainty (dimension of $\boldsymbol{h}$ and number of rows in $\boldsymbol{T}$ and $\boldsymbol{W}$)
$\mathbf{e}$	Vector of ones
$\mathbf{e}_k$	k^{th} unit basis vector
$\boldsymbol{a}_i^\top$	i^{th} row of a matrix A
$\mathbb{I}[\mathcal{E}]$	Indicator function taking a value of 1 if the expression $\mathcal{E}$ is true and 0 otherwise

7

CONCLUSIONS AND FUTURE WORK

The effective solution of vehicle routing problems can play an important role in the competitiveness, service quality and sustainability of the global economy. In this thesis, we have explored various novel models and algorithms to alleviate the traditional assumptions of deterministic parameters in vehicle routing problems arising at the operational, tactical and strategic levels of planning. In addition to these, we have also contributed methodologically to the broader field of optimization under uncertainty, by developing theory and algorithms for dynamic robust optimization problems. In the following sections, we outline the key contributions made in this thesis, as well as directions for potential future research in both vehicle routing problems as well as optimization under uncertainty.

7.1 KEY CONTRIBUTIONS (IN ORDER IN WHICH THEY APPEAR)

- We studied the modeling and solution of a broad class of vehicle routing problems under customer demand uncertainty, where the goal is to determine a single set of vehicle routes and associated fleet composition such that the total demand served on any route is less than the associated vehicle capacity, under any realization of the demands in a prespecified uncertainty set.
  - We proposed robust versions of various classical local search moves for the solution of this problem. The efficient evaluation of the local moves is enabled by closed-form expressions for the worst-case load carried by a vehicle for five classes of uncertainty sets: budget sets, factor models, ellipsoids, cardinality-constrained sets, and discrete sets. We presented data structures that allow efficient evaluation of the local moves and established time and storage complexities of updating these structures.
  - The proposed local search was shown to be modular by incorporating it within two metaheuristic implementations that determine robust feasible solutions: Iterated Local Search and Adaptive Memory Programming. The proposed metaheuristic algorithms are simple to implement and adapt: they

introduce few user-defined parameters and they do not incorporate any instance-specific features, spatiotemporal decomposition schemes, or heuristic restriction procedures to accelerate the local search process.

  - We proposed an integer programming formulation and branch-and-cut algorithm to obtain lower bounds on the optimal solution. A key feature of this formulation is a generalization of the classical rounded capacity inequalities. Efficient separation of these generalized inequalities was enabled by the same closed-form expressions and data structures that were used in the robust local search.
  - We addressed not only the classical Capacitated VRP but also all variants of the Heterogeneous and Fleet Size and Mix VRP that have been considered in the literature, as well as the Site Dependent and Multi-Depot VRP, in a single, unified framework.
  - We elucidated, via an extensive study, the computational overhead of incorporating robustness in metaheuristic algorithms, the quality of the lower bounds from the exact algorithm, as well as the tradeoff between robustness and costs for the considered uncertainty sets.

- We studied the modeling and solution of multi-period vehicle routing problems under customer order uncertainty, where the goal is to determine a minimum cost visit schedule and associated routing plan for each period of a planning horizon in which customer service requests are received dynamically during the horizon.
  - We formulated the multi-period VRP under customer order uncertainty as a multi-stage robust optimization problem. In doing so, we modeled customer orders as discrete random variables having realizations in an uncertainty set of finite (but possibly very large) cardinality. We proposed methods to construct the uncertainty set using historical data in a way that can be tuned against a user-specified risk level.
  - We conservatively approximated the multi-stage robust optimization model via a non-anticipative two-stage robust optimization model. We provided an integer programming formulation for the latter as well as a numerically efficient branch-and-cut method for its optimization.
  - We established conditions under which the solution provided by the conservative two-stage model coincides with that of the (fully adaptive) multi-stage model. In cases where these conditions are not satisfied, we derived a progressive (as opposed to conservative) approximation of the multi-stage model and presented numerical schemes for its computation.
  - We proposed algorithmic efficiencies to improve the generalized column-and-constraint generation framework for two-stage robust optimization problems with binary recourse decisions. These improvements are particularly suited in

cases when there are no second-stage costs; that is, when the recourse problem is a mere feasibility problem.

- We conducted computational experiments on test instances derived from standard benchmark datasets, and showed via out-of-sample rolling horizon simulations that robust routing plans significantly outperform traditional route plans.

- We studied the modeling and solution of consistent vehicle routing problems, in which the goal is to identify the minimum-cost set of routes that a single vehicle should follow during the multiple time periods of a planning horizon, in order to provide consistent service to a given set of customers, which we defined to be equivalent to restricting the difference between the earliest and latest vehicle arrival-times, across the multiple periods, to not exceed some given allowable limit.

  We proposed the first exact algorithms for this problem, referred to as the Consistent Traveling Salesman Problem.

  - We presented three mixed-integer linear programming formulations for this problem and introduced a new class of valid inequalities, the Inconsistent Path Elimination Constraints, to strengthen these formulations. We presented a polynomial-time algorithm to separate these inequalities in the context of a branch-and-cut framework. Further, we showed that, unlike the case of the traveling salesman problem, the Subtour Elimination Constraints and the 2-matching Constraints are not, in general, facet-defining for the underlying integer polytope of the problem.
  - In addition to the above, we developed a novel decomposition algorithm for solving these problems to guaranteed optimality. This algorithm is highly scalable, and can effectively address instances containing up to hundred customers requiring service over a five-period planning horizon.
  - We compiled a comprehensive database of benchmark instances for the Consistent Traveling Salesman problem by extending single-period TSP instances from the TSPLIB library. We calculated, for these database, the additional cost that a distributor must incur, on average, in order to implement consistent delivery schedules; that is, we quantified the "price of consistency."
  - We extended the definition of consistent vehicle routing problems to allow waiting at each customer location and to incorporate maximum route duration limits. We estimated, based on standard benchmark datasets, the savings on consistent routing costs that are to be expected by allowing the vehicle to wait at customer locations.

- We studied the strategic decision-making problem of assigning time windows to customers, referred to as the Time Window Assignment Vehicle Routing Problem. This is a two-stage stochastic optimization problem, where time window assign-

ments constitute first-stage decisions, vehicle routes adhering to the assigned time windows constitute second-stage decisions, and the objective is to minimize the expected routing costs.

  - We generalized the definition of this problem to include general scenario-based models of uncertainty in which any operational parameter may be uncertain and in which the endogenous time windows may be chosen from either continuous or discrete sets.
  - We proposed a scenario decomposition algorithm for solving this problem. The algorithm strongly outperforms all existing state of the art methods, solving fifty-four out of eighty-one previously open instances. Furthermore, it is easily parallelized, can utilize any available vehicle routing solver in a "black box" fashion, and be readily modified as a heuristic to solve large-scale instances.
  - We conducted experiments with a parallel implementation of our algorithm to solve instances consisting of up to fifteen scenarios, representing a five-fold increase compared to existing literature. We used these solutions to elucidate, via out-of-sample simulations, the cost savings that are to be expected when considering more scenarios during strategic time window assignment.

- We studied two-stage robust optimization problems with mixed discrete-continuous decisions in both stages, and presented a K-adaptability approximation that selects K candidate recourse policies before observing the realization of the uncertain parameters and that implements the best of these policies after the realization is known.

  - We established conditions under which the two-stage robust optimization problem and the K-adaptability problem are continuous, convex and tractable. We also investigated when the approximation offered by the K-adaptability problem is tight, and under which conditions the two-stage robust optimization problem and the K-adaptability problem reduce to single-stage problems.
  - We developed a branch-and-bound algorithm for the K-adaptability problem. In contrast to the existing approaches, our algorithm can handle mixed continuous and discrete decisions in both stages as well as discrete uncertainty. Furthermore, it allows for modeling continuous second-stage decisions via a novel class of highly flexible piecewise-affine decision rules.
  - We conducted extensive numerical experiments on benchmark data from various application domains. Our experiments indicated that our algorithm is highly competitive with existing state-of-the-art solution schemes.

7.2 FUTURE WORK

7.2.1 *Short Term*

1. *Static robust routing: set partitioning formulations.*

   It is well known (e.g., see [217]) that the set partitioning formulation is the most efficient integer programming formulation for the deterministic VRP. However, most mathematical programming based methods for the robust VRP have focused on vehicle flow formulations. It would be promising to extend the methods developed in Chapter 2 to the set partitioning model as the basis. A key challenge is to efficiently "robustify" the pricing routine in the associated column generation subproblem. Some early work [215] has been done towards this front, but the approach heavily capitalizes on the structure of the specific uncertainty set. It would be interesting to use our robust rounded capacity inequalities as a mechanism to enforce robustness a way that does not exploit such structure.

2. *Static robust routing: other VRP variants.*

   There is a need to generalize the methods from Chapter 2 to address problem variants with features such as simultaneous pickups and deliveries (*e.g.*, backhauls), split deliveries (where the demand can be satisfied by multiple vehicles), profits (where service is optional but results in a profit that may be imprecisely known) as well as inventory routing (where the delivery amount is a decision variable and the consumption rate is imprecisely known).

3. *Static robust routing: Out-of-sample performance of different uncertainty sets.*

   Although we proposed data-driven methods to construct each of five classes of uncertainty sets as a function of a user-specified risk level, the resulting probabilistic guarantees are *a priori* and hence, very conservative. It would be instructive to perform a systematic theoretical as well as empirical study to understand the out-of-sample performance of the routes that are robust with respect to each of the constructed uncertainty sets.

4. *Service consistency considerations: driver consistency.*

   Service consistency amounts to serving each customer over a multi-period tactical horizon using the same driver at roughly the same time of the day. The latter requirement, arrival-time consistency, was the subject of Chapter 4, where we proposed efficient algorithms to explicitly incorporate this feature. Recently, we have also been investigating efficient algorithms to explicitly incorporate the former requirement of driver consistency. The key challenge is to devise a clever decomposition (similar to the one in Chapter 4) that relies on efficient algorithms for the associated single-vehicle (multi-period traveling salesman) problem or for the single-period (multi-vehicle) routing problem.

5. *Strategic allocation of weekly visit days.*

   Chapter 5 studied the problem of allocating to customers long-term delivery time windows (within a single day). A natural generalization would be to also consider the problem of allocating weekly visit days (within a working week). Discussions with practitioners seem to indicate that the latter problem is more practically motivated in many industrial contexts as compared to the former. One reason for this is that this problem can be shown to be equivalent to the problem of partitioning the customer base among multiple depots or distribution centers, where each depot corresponds to a day of the week.

7.2.2 *Medium Term*

1. *Modeling uncertainty in travel times.*

   Existing papers that study travel time uncertainty have focused almost entirely on cardinality-constrained sets. However, this model of uncertainty is not practically motivated in case of travel times. There are several reasons for this. First, all papers ignore the underlying road network and consider a reduced graph defined only by the customer positions. This results in a highly inaccurate model since adjacent road sections (that would otherwise be reduced to a single arc in the reduced graph) can exhibit strong correlations among each other. Therefore, the assumption of independent travel times (which is implicit in the cardinality-constrained model) is difficult to justify. Second, there is a large gap between constructing travel time uncertainty sets from real traffic data.

   We believe that the factor model uncertainty set is probably a more accurate model for representing travel times. On the one hand, it allows us to represent the high-dimensional travel time uncertainty (on the actual road network) in terms of low-dimensional factors that could possibly be correlated and time-dependent (as real travel times often are). On the other hand, they can be conveniently constructed from data using well-established statistical tools like principal components or factor analysis.

2. *Static robust routing: uncertainty in other parameters.*

   Existing literature has focused almost entirely on demand uncertainty. There is a need to extend the local search and exact methods developed in Chapter 2 to address uncertainty in parameters such as travel times and service times as well (especially in view of the observation made in the previous point). This, however, is challenging since the presence of time windows would require the reinvention of efficient update rules for the robust versions of local search that have been known to work well in the deterministic versions.

3. *Simultaneous consideration of customer order and demand uncertainty.*

The multi-period vehicle routing problem studied in Chapter 3 considered uncertainty in customer requests, but assumed that their order sizes (*i.e.*, demands) are deterministic. However, in several practical applications, the demands of the customers also exhibit significant variability. The challenge is to simultaneously incorporate uncertainty in both parameters. One option is to adopt a two-phase approach: in the first phase, assume that demands are deterministic and determine a multi-period visit schedule; in the second phase, use the methods developed in Chapter 2 to determine robust single-period routes. However, it is not clear if the resulting solutions would be overly conservative.

4. *Timing considerations in multi-period vehicle routing.*

   The model of Chapter 3 allowed us to make dynamic/adaptive decisions over a multi-period horizon while ensuring that vehicle capacity constraints are always satisfied. However, timing considerations (*e.g.*, customer time windows and vehicle duration limits) where completely ignored. It is unclear how the underlying two-stage and multi-stage models can be extended to take these into account, and we believe this would constitute a valuable extension of this model.

5. *Real-time predictions in multi-period vehicle routing.*

   The framework of Chapter 3 assumed that route optimization will take place only once at the end of each day, after all orders have been received. This allowed us to use detailed models and exact solution methods to determine the routes to be executed over the next day. An interesting extension of this work would be to consider real-time decision-making in which one must decide the day (and possibly the route) in which an order will be served at the time when the order is placed *during* the day. This requires fundamentally different modeling techniques because the *degree of dynamism*; that is, the frequency at which new information is obtained and reacted upon, is significantly higher (of the order of hours and minutes, as opposed to days), thereby reducing the time available for optimization computations and, hence, the solution strategy (and, often, the solution quality).

6. *Approximation algorithms for K-adaptability problems.*

   An attractive feature of the K-adaptability formulation is its simplicity. In Chapter 6, we presented a simple iterative heuristic for the K-adaptability problem that seemed to perform extremely well for the problem instances we considered. An interesting theoretical question is to analyze the approximation guarantees of this heuristic, or perhaps, other similar heuristics, whose analysis can shed light on the structure of the K-adaptability problems.

7. *K-adaptability in two-stage distributionally robust optimization problems.*

   One interesting avenue for future work would be to extend the work of Chapter 6 to distributionally robust optimization, which is a generalization of robust optimization, in which the probability distribution governing the uncertain problem parameters is not assumed to be precisely known but rather assumed to be

contained in a so-called ambiguity set of possible distributions (*e.g.*, the set of all distributions "close" to the empirical distribution). While there has already been some work done along this direction [146], there is a need to develop analogous theoretical results and algorithms of the type developed in Chapter 6.

7.2.3 *Long Term*

1. *Dynamic robust optimization models for operational routing.*

   All existing robust optimization approaches to vehicle routing design *a priori* or static routes. The work done in Chapter 3 extended this to two-stage models in the context of a tactical planning problem. A key difference between operational and tactical level problems is that the latter affords more time for optimization, and hence, more elaborate recourse schemes. On an operational level, significantly less time is available for optimization.

   In a real time setting (*e.g.*, minutes to hours), dynamic optimization amounts to modifying the vehicle routes during their execution. This class of problems have been traditionally modeled using Markov decision processes and solved using simple, often ad-hoc, heuristics. It is unclear if these are the best approaches and if robust optimization can offer something better. Several challenges must be overcome before we can assess this.

   First, one needs to develop formulations of multi-stage robust optimization problems that appropriately capture the decision dynamics in a typical setting. For example, feasibility may often not be a concern as the applications where such problems are motivated arise in service-based routing, where services can be denied (possibly at an economic loss) but ensuring high service levels is a priority. Therefore, appropriate constraints and objectives such as maximizing throughput (*i.e.*, number of served requests) might be more appropriate. Second, these dynamic problem formulations would necessitate the development of tractable approximations of multi-stage robust optimization problems; these are still lacking in the wider robust optimization literature; however, the underlying structure in vehicle routing problems might offer a path to develop tractable schemes.

2. *Distributionally robust optimization in vehicle routing problems.*

   Distributionally robust optimization has recently started to gain a lot of attention because of its rich modeling power and underlying connections with classical problems in machine learning and statistics. A consequence is that there has been a rich arsenal of theoretical tools that have been developed to solve these problems. One broad avenue for future work would be to investigate the use of distributionally robust optimization to address vehicle routing problems under uncertainty. Some preliminary work [159, 278] has already started along this direction.

3. *K-adaptability in multi-stage robust optimization problems.*

   The scope of Chapter 6 was restricted to two-stage robust optimization problems. It would be instructive to define a K-adaptability formulation for multi-stage robust optimization problems (with more than two stages) and study its geometry and tractability. It is unclear, however, if a similar algorithm from Chapter 6 would be applicable to this setting.

BIBLIOGRAPHY

[1] E. Aarts and J. K. Lenstra, eds. *Local Search in Combinatorial Optimization*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[2] Y. Adulyasak and P. Jaillet. "Models and Algorithms for Stochastic and Robust Vehicle Routing with Deadlines." *Transportation Science* 50.2 (2016), pp. 608–626.

[3] N. Agatz, A. M. Campbell, M. Fleischmann, and M. Savelsbergh. "Challenges and Opportunities in Attended Home Delivery." In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA: Springer US, 2008, pp. 379–396.

[4] N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh. "Time Slot Management in Attended Home Delivery." *Transportation Science* 45.3 (2011), pp. 435–449.

[5] A. Agra et al. "Layered Formulation for the Robust Vehicle Routing Problem with Time Windows." In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 249–260.

[6] A. Agra et al. "The robust vehicle routing problem with time windows." *Computers & Operations Research* 40.3 (2013), pp. 856–866.

[7] A. Agra, M. Christiansen, L. M. Hvattum, and F. Rodrigues. "Robust Optimization for a Maritime Inventory Routing Problem." *Transportation Science* 52.3 (2018), pp. 509–525.

[8] M. Albareda-Sambola, E. Fernández, and G. Laporte. "The dynamic multiperiod vehicle routing problem with probabilistic information." *Computers and Operations Research* 48 (2014), pp. 31–39.

[9] G. Andreatta and G. Lulli. "A multi-period TSP with stochastic regular and urgent demands." *European Journal of Operational Research* 185.1 (2008), pp. 122–132.

[10] E. Angelelli, N. Bianchessi, R. Mansini, and M. Speranza. "Short Term Strategies for a Dynamic Multi-Period Routing Problem." *Transportation Research Part C: Emerging Technologies* 17.2 (2009), pp. 106–119.

[11] E. Angelelli, M. Grazia Speranza, and M. W. Savelsbergh. "Competitive analysis for dynamic multiperiod uncapacitated routing problems." *Networks* 49.4 (2007), pp. 308–317.

[12] E. Angelelli, M. W. Savelsbergh, and M. Grazia Speranza. "Competitive analysis of a dispatch policy for a dynamic multi-period routing problem." *Operations Research Letters* 35.6 (2007), pp. 713–721.

[13] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[14] C. Archetti, O. Jabali, and M. G. Speranza. "Multi-period Vehicle Routing Problem with Due dates." *Computers & Operations Research* 61 (2015), pp. 122–134.

[15] C. Archetti, M. G. Speranza, and D. Vigo. "Chapter 10: Vehicle Routing Problems with Profits." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 273–297.

[16] N. Ascheuer, M. Fischetti, and M. Grötschel. "A polyhedral study of the asymmetric traveling salesman problem with time windows." *Networks* 36.2 (2000), pp. 69–79.

[17] N. Ascheuer, M. Fischetti, and M. Grötschel. "Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut." *Mathematical Programming* 90.3 (2001), pp. 475–506.

[18] T. Athanasopoulos and I. Minis. "Multi-period routing in Hybrid Courier Operations." In: *Supply Chain Optimization, Design, and Management*. IGI Global, 2011, pp. 232–251.

[19] T. Athanasopoulos and I. Minis. "Efficient techniques for the multi-period vehicle routing problem with time windows within a branch and price framework." *Annals of Operations Research* 206.1 (2013), pp. 1–22.

[20] J. Ayoub and M. Poss. "Decomposition for Adjustable Robust Linear Optimization Subject to Uncertainty Polytope." *Computational Management Science* 13.2 (2016), pp. 219–239.

[21] A. Azaron and F. Kianfar. "Dynamic shortest path in stochastic dynamic networks: Ship routing problem." *European Journal of Operational Research* 144.1 (2003), pp. 138–156.

[22] N. Azi, M. Gendreau, and J.-Y. Potvin. "A dynamic vehicle routing problem with multiple delivery routes." *Annals of Operations Research* 199.1 (2012), pp. 103–112.

[23] E. K. Baker. "Technical Note—An Exact Algorithm for the Time-Constrained Traveling Salesman Problem." *Operations Research* 31.5 (1983), pp. 938–945.

[24] E. Balas. "Facets of the knapsack polytope." *Mathematical Programming* 8.1 (1975), pp. 146–164.

[25] E. Balas, M. Fischetti, and W. R. Pulleyblank. "The precedence-constrained asymmetric traveling salesman polytope." *Mathematical Programming* 68.1 (1995), pp. 241–265.

[26] E. Balas and N. Simonetti. "Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study." *INFORMS Journal on Computing* 13.1 (2001), pp. 56–75.

[27] R. Baldacci, M. Battarra, and D. Vigo. "Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs." *Networks* 54.4 (2009), pp. 178–189.

[28] R. Baldacci, N. Christofides, and A. Mingozzi. "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts." *Mathematical Programming* 115.2 (2008), pp. 351–385.

[29] R. Baldacci and A. Mingozzi. "A unified exact method for solving different classes of vehicle routing problems." *Mathematical Programming* 120 (2009), pp. 347–380.

[30] R. Baldacci, A. Mingozzi, and R. Roberti. "New route relaxation and pricing strategies for the vehicle routing problem." *Operations Research* 59.5 (2011), pp. 1269–1283.

[31] R. Baldacci, A. Mingozzi, and R. Roberti. "New State-Space Relaxations for Solving the Traveling Salesman Problem with Time Windows." *INFORMS Journal on Computing* 24.3 (2012), pp. 356–371.

[32] R. Baldacci, E. Bartolini, A. Mingozzi, and A. Valletta. "An Exact Algorithm for the Period Routing Problem." *Operations Research* 59.1 (2011), pp. 228–241.

[33] M. L. Balinski and R. E. Quandt. "On an Integer Program for a Delivery Problem." *Operations Research* 12.2 (1964), pp. 300–304.

[34] C. Bandi and D. Bertsimas. "Tractable stochastic analysis in high dimensions via robust optimization." *Mathematical programming* 134.1 (2012), pp. 23–70.

[35] T. Bektas, P. P. Repoussis, and C. D. Tarantilis. "Chapter 11: Dynamic Vehicle Routing Problems." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014, pp. 299–347.

[36] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

[37] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[38] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. "Adjustable robust solutions of uncertain linear programs." *Mathematical Programming* 99.2 (2004), pp. 351–376.

[39] R. W. Bent and P. Van Hentenryck. "Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers." *Operations Research* 52.6 (2004), pp. 977–987.

[40] G. Berbeglia, J.-F. Cordeau, and G. Laporte. "Dynamic pickup and delivery problems." *European Journal of Operational Research* 202.1 (2010), pp. 8–15.

[41] D. P. Bertsekas. *Dynamic programming and optimal control*. 4th. Vol. 1. Athena Scientific, 2017.

[42] D. Bertsimas, D. B. Brown, and C. Caramanis. "Theory and Applications of Robust Optimization." *SIAM Review* 53.3 (2011), pp. 464–501.

[43] D. Bertsimas and C. Caramanis. "Finite Adaptibility in Multistage Linear Optimization." *IEEE Transactions on Automatic Control* 55.12 (2010), pp. 2751–2766.

[44] D. Bertsimas and I. Dunning. "Multistage Robust Mixed Integer Optimization with Adaptive Partitions." *Operations Research* 64.4 (2016), pp. 980–998.

[45] D. Bertsimas and A. Georghiou. "Binary Decision Rules for Multistage Adaptive Mixed-Integer Optimization." *Available on Optimization Online* (2014).

[46] D. Bertsimas and A. Georghiou. "Design of Near Optimal Decision Rules in Multistage Adaptive Mixed-Integer Optimization." *Operations Research* 63.3 (2015), pp. 610–627.

[47] D. Bertsimas and A. Georghiou. "Binary Decision Rules for Multistage Adaptive Mixed-Integer Optimization." *Mathematical Programming* (2017), pp. 1–39.

[48] D. Bertsimas, V. Goyal, and X. A. Sun. "A Geometric Characterization of the Power of Finite Adaptability in Multistage Stochastic and Adaptive Optimization." *Mathematics of Operations Research* 36.1 (2011), pp. 24–54.

[49] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng. "Adaptive Robust Optimization for the Security Constrained Unit Commitment Problem." *IEEE Transactions on Power Systems* 28.1 (2013), pp. 52–63.

[50] D. Bertsimas. "Probabilistic combinatorial optimization problems." PhD thesis. MIT, 1988.

[51] D. Bertsimas, D. B. Brown, and C. Caramanis. "Theory and Applications of Robust Optimization." *SIAM Review* 53.3 (2011), pp. 464–501.

[52] D. Bertsimas and C. Caramanis. "Finite Adaptability in Multistage Linear Optimization." *IEEE Transactions on Automatic Control* 55.12 (2010), pp. 2751–2766.

[53] D. Bertsimas and I. Dunning. "Multistage Robust Mixed-Integer Optimization with Adaptive Partitions." *Operations Research* 64.4 (2016), pp. 980–998.

[54] D. Bertsimas and A. Georghiou. "Design of Near Optimal Decision Rules in Multistage Adaptive Mixed-Integer Optimization." *Operations Research* 63.3 (2015), pp. 610–627.

[55] D. Bertsimas, S. Gupta, and J. Tay. "Scalable Robust and Adaptive Inventory Routing." *Available on Optimization Online* (2016).

[56] D. Bertsimas, V. Gupta, and N. Kallus. "Data-driven robust optimization." *Mathematical Programming* 167.2 (2018), pp. 235–292.

[57] D. Bertsimas and M. Sim. "The price of robustness." *Operations Research* 52.1 (2004), pp. 35–53.

[58] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer New York, 2011.

[59] J. W. Blankenship and J. E. Falk. "Infinitely constrained optimization problems." *Journal of Optimization Theory and Applications* 19.2 (1976), pp. 261–281.

[60] C. Blum and A. Roli. "Metaheuristics in combinatorial optimization." *ACM Computing Surveys* 35.3 (2003), pp. 268–308.

[61] N. Bostel, P. Dejax, P. Guez, and F. Tricoire. "Multiperiod Planning and Routing on a Rolling Horizon for Field Force Optimization Logistics." In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA: Springer US, 2008, pp. 503–525.

[62] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[63] S. Braaten, O. Gjønnes, L. M. Hvattum, and G. Tirado. "Heuristics for the robust vehicle routing problem with time windows." *Expert Systems with Applications* 77 (2017), pp. 136–147.

[64] O. Bräysy. "A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows." *INFORMS Journal on Computing* 15.4 (2003), pp. 347–368.

[65] O. Bräysy and M. Gendreau. "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms." *Transportation Science* 39.1 (2005), pp. 104–118.

[66] O. Bräysy and G. Hasle. "Chapter 12: Software Tools and Emerging Technologies for Vehicle Routing and Intermodal Transportation." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014, pp. 351–380.

[67] B. P. Bruck, J.-F. Cordeau, and M. Iori. "A practical time slot management and routing problem for attended home services." *Omega* (2017).

[68] C. Buchheim and J. Kurtz. "Min-Max-Min Robustness: A New Approach to Combinatorial Optimization under Uncertainty based on Multiple Solutions." *Electronic Notes in Discrete Mathematics* 52 (2016), pp. 45–52.

[69] C. Buchheim and J. Kurtz. "Min–max–min Robust Combinatorial Optimization." *Mathematical Programming* 163.1 (2017), pp. 1–23.

[70] C. Buchheim and J. Kurtz. "Complexity of min–max–min robustness for combinatorial optimization under discrete uncertainty." *Discrete Optimization* 28.1 (2018), pp. 1–15.

[71] A. M. Campbell, L. W. Clarke, and M. W. P. Savelsbergh. "Inventory Routing in Practice." In: *The Vehicle Routing Problem*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2002, pp. 309–330.

[72] A. M. Campbell, M. Gendreau, and B. W. Thomas. "The orienteering problem with stochastic travel and service times." *Annals of Operations Research* 186.1 (2011), pp. 61–81.

[73] A. M. Campbell and B. W. Thomas. "Challenges and Advances in A Priori Routing." In: *Operations Research/Computer Science Interfaces*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Springer US, 2008, pp. 123–142.

[74] A. M. Campbell and J. H. Wilson. "Forty years of periodic vehicle routing." *Networks* 63.1 (2014), pp. 2–15.

[75] E. Cao, M. Lai, and H. Yang. "Open vehicle routing problem with demand uncertainty and its robust strategies." *Expert Systems with Applications* 41.7 (2014), pp. 3569–3575.

[76] S. Ceria and R. A. Stubbs. "Incorporating estimation errors into portfolio selection: Robust portfolio construction." *Journal of Asset Management* 7.2 (2006), pp. 109–127.

[77] I.-M. Chao, B. Golden, and E. Wasil. "A Computational Study Of A New Heuristic For The Site-Dependent Vehicle Routing Problem." *INFOR: Information Systems and Operational Research* 37.3 (1999), pp. 319–336.

[78] L. Chen, M. Gendreau, M. H. Hà, and A. Langevin. "A robust optimization approach for the road network daily maintenance routing problem with uncertain service time." *Transportation Research Part E: Logistics and Transportation Review* 85 (2016), pp. 40–51.

[79] X. Chen and Y. Zhang. "Uncertain Linear Programs: Extended Affinely Adjustable Robust Counterparts." *Operations Research* 57.6 (2009), pp. 1469–1482.

[80] L. Cheng and M. A. Duran. "Logistics for world-wide crude oil transportation using discrete event simulation and optimal control." *Computers & Chemical Engineering* 28.6-7 (2004), pp. 897–911.

[81] E. Choi and D.-W. Tcha. "A column generation approach to the heterogeneous fleet vehicle routing problem." *Computers & Operations Research* 34.7 (2007), pp. 2080 –2095.

[82] M. Christiansen and K. Fagerholt. "Chapter 13: Ship Routing and Scheduling in Industrial and Tramp Shipping." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 381–408.

[83] N. Christofides, A. Mingozzi, and P. Toth. "State-space relaxation procedures for the computation of bounds to routing problems." *Networks* 11.2 (1981), pp. 145–164.

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

[85] L. C. Coelho, J.-F. Cordeau, and G. Laporte. "Thirty Years of Inventory Routing." *Transportation Science* 48.1 (2014), pp. 1–19.

[86] M. Conforti, G. Cornuejols, and G. Zambelli. *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[87] J.-F. Cordeau, M. Gendreau, and G. Laporte. "A tabu search heuristic for periodic and multi-depot vehicle routing problems." *Networks* 30.2 (1997), pp. 105–119.

[88] J.-F. Cordeau and G. Laporte. "A Tabu Search Algorithm For The Site Dependent Vehicle Routing Problem With Time Windows." *INFOR: Information Systems and Operational Research* 39.3 (2001), pp. 292–298.

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

[90] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009.

[91] T. G. Crainic, M. Gendreau, and P. Dejax. "Dynamic and Stochastic Models for the Allocation of Empty Containers." *Operations Research* 41.1 (1993), pp. 102–126.

[92] Éric D Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin. "Adaptive memory programming: A unified view of metaheuristics." *European Journal of Operational Research* 135.1 (2001), pp. 1 –16.

[93] C. F. Daganzo. "The length of tours in zones of different shapes." *Transportation Research Part B: Methodological* 18.2 (1984), pp. 135–145.

[94] K. Dalmeijer and R. Spliet. "A branch-and-cut algorithm for the Time Window Assignment Vehicle Routing Problem." *Computers & Operations Research* 89 (2018), pp. 140–152.

[95] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem." *Management Science* 6.1 (1959), pp. 80–91.

[96] G. Dantzig, R Fulkerson, and S Johnson. "Solution of a large-scale traveling-salesman problem." *Journal of the Operations Research Society of America* 2.4 (Nov. 1954), pp. 393–410.

[97] G. B. Dantzig. "Discrete-variable extremum problems." *Operations Research* 5 (1957), pp. 266–277.

[98] S. Dash, O. Günlük, A. Lodi, and A. Tramontani. "A Time Bucket Formulation for the Traveling Salesman Problem with Time Windows." *INFORMS Journal on Computing* 24.1 (2012), pp. 132–147.

[99] J. M. Day, P. D. Wright, T. Schoenherr, M. Venkataramanan, and K. Gaudette. "Improving routing and scheduling decisions at a distributor of industrial gasses." *Omega* 37.1 (2009), pp. 227 –237.

[100] I. Dayarian, T. G. Crainic, M. Gendreau, and W. Rei. "A branch-and-price approach for a multi-period vehicle routing problem." *Computers & Operations Research* 55 (2015), pp. 167–184.

[101] S. T. DeNegre and T. K. Ralphs. "A Branch-and-cut Algorithm for Integer Bilevel Linear Programs." In: *Operations Research and Cyber-Infrastructure*. Ed. by J. W. Chinneck, B. Kristjansson, and M. J. Saltzman. Boston, MA: Springer US, 2009, pp. 65–78.

[102] G. Desaulniers, O. B. Madsen, and S. Ropke. "Chapter 5: The Vehicle Routing Problem with Time Windows." In: *Vehicle Routing: Problems, Methods and Applications*. Ed. by P. Toth and D. Vigo. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014, pp. 119–159.

[103] M. Desrochers and G. Laporte. "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints." *Operations Research Letters* 1.February (1991), pp. 27–36.

[104] K. F. Doerner and J.-J. Salazar-González. "Chapter 7: Pickup-and-Delivery Problems for People Transportation." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 193–212.

[105] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles." *Mathematical Programming* 91.2 (Jan. 2002), pp. 201–213.

[106] M. Drexl. "Synchronization in Vehicle Routing – A Survey of VRPs with Multiple Synchronization Constraints." *Transportation Science* 46.3 (2012), pp. 297–316.

[107] M. Dror, G. Laporte, and P. Trudeau. "Vehicle routing with stochastic demands: Properties and solution frameworks." *Transportation science* 23.3 (1989), pp. 166–176.

[108] Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon. "An Optimal Algorithm for the Traveling Salesman Problem with Time Windows." *Operations Research* 43.2 (1995), pp. 367–371.

[109] A. L. Erera, J. C. Morales, and M. Savelsbergh. "The Vehicle Routing Problem with Stochastic Demand and Duration Constraints." *Transportation Science* 44.4 (2010), pp. 474–492.

[110] D. Feillet, T. Garaix, F. Lehuédé, O. Péton, and D. Quadri. "A new consistent vehicle routing problem for the transportation of people with disabilities." *Networks* 63.3 (2014), pp. 211–224.

[111] M. A. Figliozzi. "Planning approximations to the average length of vehicle routing problems with time window constraints." *Transportation Research Part B: Methodological* 43.4 (2009), pp. 438–447.

[112] M. Fischetti. "Facets of the Asymmetric Traveling Salesman Polytope." *Mathematics of Operations Research* 16.1 (Feb. 1991), pp. 42–56.

[113] M. Fischetti, J. J. S. González, and P. Toth. "Solving the Orienteering Problem through Branch-and-Cut." *INFORMS Journal on Computing* 10.2 (1998), pp. 133–148.

[114] M. Fischetti, A. Lodi, and P. Toth. "Exact Methods for the Asymmetric Traveling Salesman Problem." English. In: *The Traveling Salesman Problem and Its Variations*. Ed. by G. Gutin and A. P. Punnen. Vol. 12. Combinatorial Optimization. Springer US, 2007, pp. 169–205.

[115] M. Fischetti and P. Toth. "A Polyhedral Approach to the Asymmetric Traveling Salesman Problem." *Management Science* 43.11 (Nov. 1997), pp. 1520–1536.

[116] T. Flatberg, G. Hasle, O. Kloster, E. J. Nilssen, and A. Riise. "Dynamic And Stochastic Vehicle Routing In Practice." In: *Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies*. Ed. by V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, and I. Minis. Boston, MA: Springer US, 2007, pp. 41–63.

[117] F. Focacci, A. Lodi, and M. Milano. "A Hybrid Exact Algorithm for the TSPTW." *INFORMS Journal on Computing* 14.4 (2002), pp. 403–417.

[118] P. M. Francis, K. R. Smilowitz, and M. Tzur. "The Period Vehicle Routing Problem and its Extensions." In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA: Springer US, 2008, pp. 73–102.

[119] R. Fukasawa et al. "Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem." *Mathematical Programming* 106.3 (2006), pp. 491–511.

[120] B. Funke, T. Grünert, and S. Irnich. "Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration." *Journal of Heuristics* 11.4 (2005), pp. 267–306.

[121] V. Gabrel, C. Murat, and A. Thiele. "Recent Advances in Robust Optimization: An Overview." *European Journal of Operational Research* 235.3 (2014), pp. 471–483.

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

[123] C. Gauvin, G. Desaulniers, and M. Gendreau. "A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands." *Computers & Operations Research* 50 (2014), pp. 141–153.

[124] B. Gavish and S. C. Graves. *The travelling salesman problem and related problems*. Working Paper OR 078–78, Operations Research Center, Massachusetts Institute of Technology. 1978.

[125] M. Gendreau, O. Jabali, and W. Rei. "Chapter 8: Stochastic Vehicle Routing Problems." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 213–239.

[126] M. Gendreau, G. Laporte, and R. Séguin. "An Exact Algorithm for the Vehicle Routing Problem with Stochastic Demands and Customers." *Transportation Science* 29.2 (1995), pp. 143–155.

[127] M. Gendreau, G. Laporte, and F. Semet. "Heuristics and lower bounds for the bin packing problem with conflicts." *Computers & Operations Research* 31.3 (2004), pp. 347–358.

[128] A. Georghiou, W. Wiesemann, and D. Kuhn. "Generalized Decision Rule Approximations for Stochastic Programming via Liftings." *Mathematical Programming* 152.1 (2015), pp. 301–338.

[129] F. Glover. "Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges." In: *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*. Ed. by R. S. Barr, R. V. Helgason, and J. L. Kennington. Boston, MA: Springer US, 1997, pp. 1–75.

[130] M. T. Godinho, L. Gouveia, and P. Pesneau. "On a time-dependent formulation and an updated classification of ATSP formulations." In: *Progress in combinatorial optimization*. Ed. by A. R. Mahjoub. Wiley, 2011, pp. 251–305.

[131] J. Goh and M. Sim. "Distributionally Robust Optimization and its Tractable Approximations." *Operations Research* 58.4 (2010), pp. 902–917.

[132] B. L. Golden, A. A. Assad, and E. A. Wasil. "Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries." In: *The Vehicle Routing Problem*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2002. Chap. 10, pp. 245–286.

[133] B. Golden, S. Raghavan, and E. Wasil, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces. Boston, MA: Springer US, 2008.

[134] B. Golden, A. Assad, L. Levy, and F. Gheysens. "The fleet size and mix vehicle routing problem." *Computers & Operations Research* 11.1 (1984), pp. 49 –66.

[135] B. L. Gorissen, I. Yanıkoğlu, and D. den Hertog. "A practical guide to robust optimization." *Omega* 53 (2015), pp. 124–137.

[136] B. L. Gorissen and D. den Hertog. "Robust counterparts of inequalities containing sums of maxima of linear functions." *European Journal of Operational Research* 227.1 (2013), pp. 30–43.

[137] C. E. Gounaris, W. Wiesemann, and C. A. Floudas. "The Robust Capacitated Vehicle Routing Problem Under Demand Uncertainty." *Operations Research* 61.3 (2013), pp. 677–693.

[138] C. E. Gounaris, P. P. Repoussis, C. D. Tarantilis, W. Wiesemann, and C. A. Floudas. "An Adaptive Memory Programming Framework for the Robust Capacitated Vehicle Routing Problem." *Transportation Science* 50.4 (2016), pp. 1239–1260.

[139] L. Gouveia. "A result on projection for the vehicle routing problem." *European Journal of Operational Research* 85.3 (1995), pp. 610 –624.

[140] C. Groër, B. Golden, and E. Wasil. "The Consistent Vehicle Routing Problem." *Manufacturing & Service Operations Management* 11.4 (Oct. 2009), pp. 630–643.

[141] P.-O. Groß, M. W. Ulmer, J. F. Ehmke, and D. C. Mattfeld. "Exploiting Travel Time Information for Reliable Routing in City Logistics." *Transportation Research Procedia* 10 (2015), pp. 652–661.

[142] M. Grötschel and M. W. Padberg. "Polyhedral Theory." In: *The Travelling Salesman Problem*. Ed. by E. Lawler, J. Lenstra, A. R. Kan, and D. Shmoys. New York: John Wiley and Sons, 1985, pp. 251–305.

[143] E. Guslitser. "Uncertainty-Immunized Solutions in Linear Programming." MA thesis. Technion, Israeli Institute of Technology, 2002.

[144] E. Hadjiconstantinou and D. Roberts. "Routing under Uncertainty: An Application in the Scheduling of Field Service Engineers." In: *The Vehicle Routing Problem*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2002. Chap. 13, pp. 331–352.

[145] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. "*K*-Adaptability in Two-Stage Robust Binary Programming." *Operations Research* 63.4 (2015), pp. 877–891.

[146] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. "K-adaptability in two-stage distributionally robust binary programming." *Operations Research Letters* 44.1 (2016), pp. 6 –11.

[147] D. Haugland, S. C. Ho, and G. Laporte. "Designing delivery districts for the vehicle routing problem with stochastic demands." *European Journal of Operational Research* 180.3 (2007), pp. 997–1010.

[148] K. Helsgaun. "An effective implementation of the Lin–Kernighan traveling salesman heuristic." *European Journal of Operational Research* 126.1 (2000), pp. 106 – 130.

[149] V. Hemmelmayr, K. F. Doerner, R. F. Hartl, and M. W. P. Savelsbergh. "Delivery strategies for blood products supplies." *OR Spectrum* 31.4 (2009), pp. 707–725.

[150] F. Hernandez, M. Gendreau, and J.-Y. Potvin. "Heuristics for tactical time slot management: a periodic vehicle routing problem view." *International Transactions in Operational Research* 24.6 (2017), pp. 1233–1252.

[151] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen. "Industrial aspects and literature survey: Fleet composition and routing." *Computers & Operations Research* 37.12 (2010), pp. 2041 –2061.

[152] P. Horner. "Innovation powers dynamic VR sector." *OR/MS-Today* 45.1 (2018), pp. 42–45.

[153] C. Hu, J. Lu, X. Liu, and G. Zhang. "Robust vehicle routing problem with hard time windows under demand and travel time uncertainty." *Computers & Operations Research* 94 (2018), pp. 139–153.

[154] IBM. *ILOG CPLEX Optimizer*. [Online; accessed 27-July-2018]. 2018.

[155] I. Ioachim, J. Desrosiers, F. Soumis, and N. Bélanger. "Fleet assignment and routing with schedule synchronization constraints." *European Journal of Operational Research* 119.1 (1999), pp. 75 –90.

[156] S. Irnich, M. Schneider, and D. Vigo. "Chapter 9: Four Variants of the Vehicle Routing Problem." In: *Vehicle Routing: Problems, Methods and Applications*. Ed. by P. Toth and D. Vigo. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014, pp. 241–271.

[157] O. Jabali, R. Leus, T. van Woensel, and T. de Kok. "Self-imposed time windows in vehicle routing problems." *OR Spectrum* 37.2 (2015), pp. 331–352.

[158] P. Jaillet. "A Priori Solution of a Traveling Salesman Problem in Which a Random Subset of the Customers Are Visited." *Operations Research* 36.6 (1988), pp. 929–936.

[159] P. Jaillet, J. Qi, and M. Sim. "Routing Optimization Under Uncertainty." *Operations Research* 64.1 (2016), pp. 186–200.

[160] P. Jaillet and M. R. Wagner. "Online Vehicle Routing Problems: A Survey." In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA: Springer US, 2008, pp. 221–237.

[161] K. Jansen and S. Öhring. "Approximation Algorithms for Time Constrained Scheduling." *Information and Computation* 132.2 (1997), pp. 85–108.

[162] Jean-François Cordeau and Gilbert Laporte and Martin W.P. Savelsbergh and Daniele Vigo. "Chapter 6 Vehicle Routing." In: *Handbooks in Operations Research and Management Science*. Ed. by Cynthia Barnhart and Gilbert Laporte. Vol. 14. Elsevier, 2007, pp. 367–428.

[163] R. Jiang, M. Zhang, G. Li, and Y. Guan. "Two-Stage Robust Power Grid Optimization Problem." Available on Optimization Online. 2010.

[164] B. Kallehauge, N. Boland, and O. B. Madsen. "Path inequalities for the vehicle routing problem with time windows." *Networks* 49.4 (2007), pp. 273–293.

[165] I. Kara, O. N. Koc, F. Altıparmak, and B. Dengiz. "New integer linear programming formulation for the traveling salesman problem with time windows: minimizing tour duration with waiting times." *Optimization* 62.10 (2013), pp. 1309–1319.

[166] R. M. Karp. "Reducibility among Combinatorial Problems." In: *Complexity of Computer Computations*. Ed. by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103.

[167] A. Kearney. *28th Annual State of Logistics Report*. Tech. rep. Council of Supply Chain Management Professionals (CSCMP), 2017.

[168] J. E. Kelley. "The Cutting-Plane Method for Solving Convex Programs." *Journal of the Society for Industrial and Applied Mathematics* 8.4 (1960), pp. 703–712.

[169] M. A. Klapp, A. L. Erera, and A. Toriello. "The Dynamic Dispatch Waves Problem for Same-Day Delivery." *Available on Optimization Online* (2016).

[170] Çağrı Koç, T. Bektaş, O. Jabali, and G. Laporte. "Thirty years of heterogeneous vehicle routing." *European Journal of Operational Research* 249.1 (2016), pp. 1 –21.

[171] A. A. Kovacs, S. N. Parragh, and R. F. Hartl. "A template-based adaptive large neighborhood search for the consistent vehicle routing problem." *Networks* 63.1 (2014), pp. 60–81.

[172] A. A. Kovacs, B. L. Golden, R. F. Hartl, and S. N. Parragh. "Vehicle routing problems in which consistency considerations are important: A survey." *Networks* 64.3 (2014), pp. 192–213.

[173] A. A. Kovacs, B. L. Golden, R. F. Hartl, and S. N. Parragh. "The Generalized Consistent Vehicle Routing Problem." *Transportation Science* 49.4 (2015), pp. 796–816.

[174] D. Kuhn, W. Wiesemann, and A. Georghiou. "Primal and Dual Linear Decision Rules in Stochastic and Robust Optimization." *Mathematical Programming* 130.1 (2011), pp. 177–209.

[175] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas, and F. Soumis. "A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows." *Networks* 23.7 (1993), pp. 631–640.

[176] G. Laporte. "Fifty Years of Vehicle Routing." *Transportation Science* 43.4 (2009), pp. 408–416.

[177] G. Laporte, F. V. Louveaux, and L. van Hamme. "An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands." *Operations Research* 50.3 (2002), pp. 415–423.

[178] G. Laporte, F. V. Louveaux, and H. Mercure. "A Priori Optimization of the Probabilistic Traveling Salesman Problem." *Operations Research* 42.3 (1994), pp. 543–549.

[179] G. Laporte, F. Louveaux, and H. Mercure. "Models and exact solutions for a class of stochastic location-routing problems." *European Journal of Operational Research* 39 (1989), pp. 71–78.

[180] G. Laporte, H. Mercure, and Y. Nobert. "An exact algorithm for the asymmetrical capacitated vehicle routing problem." *Networks* 16.1 (1986), pp. 33–46.

[181] G. Laporte, Y. Nobert, and M. Desrochers. "Optimal Routing under Capacity and Distance Restrictions." *Operations Research* 33.5 (1985), pp. 1050–1073.

[182] G. Laporte, S. Ropke, and T. Vidal. "Chapter 4: Heuristics for the Vehicle Routing Problem." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 87–116.

[183] A Larsen, O Madsen, and M Solomon. "Partially dynamic vehicle routing—models and algorithms." *Journal of the Operational Research Society* 53.6 (2002), pp. 637–646.

[184] A. Larsen, O. B. Madsen, and M. M. Solomon. "Recent Developments in Dynamic Vehicle Routing Systems." In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA: Springer US, 2008, pp. 199–218.

[185] C Lee, K Lee, and S Park. "Robust vehicle routing problem with deadlines and travel time/demand uncertainty." *Journal of the Operational Research Society* 63.9 (2012), pp. 1294–1306.

[186] A. N. Letchford, G. Reinelt, and D. O. Theis. "A Faster Exact Separation Algorithm for Blossom Inequalities." English. In: *Integer Programming and Combinatorial Optimization*. Ed. by D. Bienstock and G. Nemhauser. Vol. 3064. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 196–205.

[187] J.-Q. Li. *A Computational Study of Bi-directional Dynamic Programming for the Traveling Salesman Problem with Time Windows*. Working Paper, University of California, Berkeley. 2009.

[188] J.-Q. Li, D. Borenstein, and P. B. Mirchandani. "A decision support system for the single-depot vehicle rescheduling problem." *Computers & Operations Research* 34.4 (2007), pp. 1008–1032.

[189] J.-R. Lin and T.-H. Yang. "Strategic design of public bicycle sharing systems with service level constraints." *Transportation Research Part E: Logistics and Transportation Review* 47.2 (2011), pp. 284–294.

[190] S. Lin and B. W. Kernighan. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem." *Operations Research* 21.2 (1973), pp. 498–516.

[191] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated Local Search." In: *Handbook of Metaheuristics*. Ed. by F. Glover and G. A. Kochenberger. Boston, MA: Springer US, 2003, pp. 320–353.

[192] Z. Luo, H. Qin, C. Che, and A. Lim". "On service consistency in multi-period vehicle routing." *European Journal of Operational Research* 243.3 (2015), pp. 731 –744.

[193] J. Lysgaard, A. N. Letchford, and R. W. Eglese. "A new branch-and-cut algorithm for the capacitated vehicle routing problem." *Mathematical Programming* 100.2 (2004), pp. 423–445.

[194] F. Maffioli and A. Sciomachen. "A mixed-integer model for solving ordering problems with side constraints." *Annals of Operations Research* 69 (1997), pp. 277–297.

[195] C. Malandraki and M. S. Daskin. "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms." *Transportation Science* 26.3 (1992), pp. 185–200.

[196] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[197] S. T. McCormick, M. Rao, and G. Rinaldi. "Easy and difficult objective functions for max cut." *Mathematical Programming* 94.2 (2003), pp. 459–466.

[198] C. E. Miller, A. W. Tucker, and R. A. Zemlin. "Integer Programming Formulation of Traveling Salesman Problems." *Journal of the ACM* 7.4 (Oct. 1960), pp. 326–329.

[199] A. Mingozzi, L. Bianco, and S. Ricciardelli. "Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints." *Operations Research* 45.3 (1997), pp. 365–377.

[200] I. Minis, K. Mamasis, and V. Zeimpekis. "Real-time management of vehicle breakdowns in urban freight distribution." *Journal of Heuristics* 18.3 (2011), pp. 375–400.

[201] Q Mu, Z Fu, J Lysgaard, and R Eglese. "Disruption management of the vehicle routing problem with vehicle breakdown." *Journal of the Operational Research Society* 62.4 (2011), pp. 742–749.

[202] P. Munari et al. *The robust vehicle routing problem with time windows: compact formulation and branch-price-and-cut method*. Tech. rep. Federal University of Sao Carlos, Brazil, 2018.

[203] A. Mutapcic and S. Boyd. "Cutting-set methods for robust convex optimization with pessimizing oracles." *Optimization Methods and Software* 24.3 (2009), pp. 381–406.

[204] B. Nag, B. L. Golden, and A. A. Assad. "Vehicle routing with site dependencies." In: *Vehicle Routing: Methods and Studies*. Ed. by B. Golden and A. Assad. Amsterdam: Elsevier, 1988, pp. 149–159.

[205] T. Nuortio, J. Kytojoki, H. Niska, and O. Bräysy. "Improved route planning and scheduling of waste collection and transport." *Expert Systems with Applications* 30.2 (2006), pp. 223–232.

[206] T. Öncan, I. K. Altınel, and G. Laporte. "A comparative analysis of several asymmetric traveling salesman problem formulations." *Computers & Operations Research* 36.3 (2009), pp. 637–654.

[207] F. Ordóñez. "Robust Vehicle Routing." *INFORMS TutORials in Operations Research* Risk and Optimization in an Uncertain World (2010), pp. 153–178.

[208] M. Padberg and G. Rinaldi. "An efficient algorithm for the minimum capacity cut problem." *Mathematical Programming* 47.1-3 (May 1990), pp. 19–36.

[209] M. W. Padberg and M. Rao. "Odd minimum cut-sets and *b*-matchings." *Mathematics of Operations Research* 7.1 (1982), pp. 67 –80.

[210] D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa. "Improved branch-cut-and-price for capacitated vehicle routing." *Mathematical Programming Computation* 9.1 (2017), pp. 61–100.

[211] P. H. V. Penna, A. Subramanian, L. S. Ochi, T. Vidal, and C. Prins. "A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet." *Annals of Operations Research* (2017), pp. 1–70.

[212] G. Pesant, M. Gendreau, J.-Y. Potvin, and J.-M. Rousseau. "An Exact Constraint Logic Programming Algorithm for the Traveling Salesman Problem with Time Windows." *Transportation Science* 32.1 (1998), pp. 12–29.

[213] A. Pessoa, M. P. De Aragão, and E. Uchoa. "Robust branch-cut-and-price algorithms for vehicle routing problems." *The vehicle routing problem: Latest advances and new challenges* 43 (2008), pp. 297–325.

[214] A. Pessoa, R. Sadykov, and E. Uchoa. "Enhanced Branch-Cut-and-Price algorithm for heterogeneous fleet vehicle routing problems." *European Journal of Operational Research* 270.2 (2018), pp. 530 –543.

[215] A. Pessoa, M. Poss, R. Sadykov, and F. Vanderbeck. "Solving the robust CVRP under demand uncertainty." In: *ODYSSEUS 2018 - 7th International Workshop on Freight Transportation and Logistics*. Calgliari, Italy, June 2018.

[216] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. "A review of dynamic vehicle routing problems." *European Journal of Operational Research* 225.1 (2013), pp. 1–11.

[217] M. Poggi and E. Uchoa. "Chapter 3: New Exact Algorithms for the Capacitated Vehicle Routing Problem." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 59–86.

[218] K. Postek and D. den Hertog. "Multistage Adjustable Robust Mixed-Integer Optimization via Iterative Splitting of the Uncertainty Set." *INFORMS Journal on Computing* 28.3 (2016), pp. 553–574.

[219] W. B. Powell, Y. Sheffi, K. S. Nickerson, K. Butterbaugh, and S. Atherton. "Maximizing Profits for North American Van Lines Truckload Division: A New Framework for Pricing and Operations." *Interfaces* 18.1 (1988), pp. 21–41.

[220] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[221] G. Reinelt. "TSPLIB – A Traveling Salesman Problem Library." *ORSA Journal on Computing* 3.4 (1991), pp. 376–384.

[222] R. Roberti and P. Toth. "Models and algorithms for the Asymmetric Traveling Salesman Problem: an experimental comparison." English. *EURO Journal on Transportation and Logistics* 1.1-2 (2012), pp. 113–133.

[223] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. 3rd. Springer, 2009.

[224] M. Schilde, K. Doerner, and R. Hartl. "Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem." *European Journal of Operational Research* 238.1 (2014), pp. 18–30.

[225] M. Schneider, A. Stenger, F. Schwahn, and D. Vigo. "Territory-Based Vehicle Routing in the Presence of Time-Window Constraints." *Transportation Science* 49.4 (2015), pp. 732–751.

[226] N. Secomandi. "A rollout policy for the vehicle routing problem with stochastic demands." *Operations Research* 49 (2001), pp. 796–802.

[227] N. Secomandi and F. Margot. "Reoptimization Approaches for the Vehicle-Routing Problem with Stochastic Demands." *Operations Research* 57.1 (2009), pp. 214–230.

[228] F. Semet, P. Toth, and D. Vigo. "Chapter 2: Classical Exact Algorithms for the Capacitated Vehicle Routing Problem." In: *Vehicle Routing*. Ed. by P. Toth and D. Vigo. Society for Industrial and Applied Mathematics, 2014, pp. 37–57.

[229] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming*. Society for Industrial and Applied Mathematics, 2009.

[230] Z. Shen, M. M. Dessouky, and F. Ordóñez. "A two-stage vehicle routing model for large-scale bioterrorism emergencies." *Networks* 54.4 (2009), pp. 255–269.

[231] J. Siek, L.-Q. Lee, and A. Lumsdaine. *Boost Graph Library*. Available at: http://www.boost.org/libs/graph/ [Accessed 12th December, 2014]. 2000.

[232] G. Skorobohatyj. *Finding a Minimum Cut between all Pairs of Nodes in an Undirected Graph*. Available at: http://elib.zib.de/pub/Packages/mathprog/mincut/index.html [Accessed 12th December, 2014]. 2004.

[233] K. Smilowitz, M. Nowak, and T. Jiang. "Workforce Management in Periodic Delivery Operations." *Transportation Science* 47.2 (2013), pp. 214–230.

[234] E. L. Solano-Charris, C. Prins, and A. C. Santos. "A Robust optimization approach for the Vehicle Routing problem with uncertain travel cost." In: *2014 International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2014.

[235] E. L. Solano-Charris, C. Prins, and A. C. Santos. "Heuristic Approaches for the Robust Vehicle Routing Problem." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 384–395.

[236] E. L. Solano-Charris, C. Prins, and A. C. Santos. "Solving the bi-objective Robust Vehicle Routing Problem with uncertain costs and demands." *RAIRO - Operations Research* 50.4-5 (2016), pp. 689–714.

[237] E. Solano-Charris, C. Prins, and A. C. Santos. "Local search based metaheuristics for the robust vehicle routing problem with discrete scenarios." *Applied Soft Computing* 32 (2015), pp. 518–531.

[238] M. M. Solomon. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." *Operations Research* 35.2 (1987), pp. 254 –265.

[239] O. Solyalı, J.-F. Cordeau, and G. Laporte. "Robust Inventory Routing Under Demand Uncertainty." *Transportation Science* 46.3 (2012), pp. 327–340.

[240] R. Spliet, S. Dabia, and T. van Woensel. "The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times." *Transportation Science* 52.2 (2018), pp. 261–276.

[241] R. Spliet and G. Desaulniers. "The discrete time window assignment vehicle routing problem." *European Journal of Operational Research* 244.2 (2015), pp. 379–391.

[242] R. Spliet and A. F. Gabor. "The Time Window Assignment Vehicle Routing Problem." *Transportation Science* 49.4 (2015), pp. 721–731.

[243] L. Sun and B. Wang. "A Goal-Robust-Optimization Approach for Solving Open Vehicle Routing Problems with Demand Uncertainty." *Wireless Personal Communications* (2018).

[244] I. Sungur, F. Ordóñez, and M. Dessouky. "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty." *IIE Transactions* 40.5 (2008), pp. 509–523.

[245] I. Sungur, Y. Ren, F. Ordóñez, M. Dessouky, and H. Zhong. "A Model and Algorithm for the Courier Delivery Problem with Uncertainty." *Transportation Science* 44.2 (2010), pp. 193–205.

[246] Taillard, É. D. "A heuristic column generation method for the heterogeneous fleet VRP." *RAIRO Recherche Opérationnelle* 33.1 (1999), pp. 1–14.

[247] H. Tang, E. Miller-Hooks, and R. Tomastik. "Scheduling technicians for planned maintenance of geographically distributed equipment." *Transportation Research Part E: Logistics and Transportation Review* 43.5 (2007), pp. 591–609.

[248] Y. Tang, J.-P. P. Richard, and J. C. Smith. "A class of algorithms for mixed-integer bilevel min–max optimization." *Journal of Global Optimization* 66.2 (2016), pp. 225–262.

[249] C. Tarantilis, F. Stavropoulou, and P. Repoussis. "A template-based Tabu Search algorithm for the Consistent Vehicle Routing Problem." *Expert Systems with Applications* 39.4 (2012), pp. 4233–4239.

[250] D. Teodorović, E. Krčmar-Nožić, and G. Pavkoviç. "The mixed fleet stochastic vehicle routing problem." *Transportation Planning and Technology* 19.1 (1995), pp. 31–43.

[251] A. Thiele, T. Terry, and M. Epelman. *Robust Linear Optimization with Recourse*. Tech. rep. Lehigh University and University of Michigan, 2010.

[252] C. Tilk and S. Irnich. "Dynamic Programming for the Minimum Tour Duration Problem." *Transportation Science* 51.2 (2017), pp. 549–565.

[253] G. Tirado, L. M. Hvattum, K. Fagerholt, and J.-F. Cordeau. "Heuristics for dynamic and stochastic routing in industrial shipping." *Computers & Operations Research* 40.1 (2013), pp. 253–263.

[254] N. E. Toklu, L. M. Gambardella, and R. Montemanni. "A Multiple Ant Colony System for a Vehicle Routing Problem with Time Windows and Uncertain Travel Times." *Journal of Traffic and Logistics Engineering* 2.1 (2014), pp. 52–58.

[255] P. Toth and D. Vigo, eds. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

[256] P. Toth and D. Vigo, eds. *Vehicle Routing: Problems, Methods and Applications, Second Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.

[257] F. Tricoire. "Vehicle and personnel routing optimization in the service sector: application to water distribution and treatment." *4OR* 5.2 (2007), pp. 165–168.

[258] M. W. Ulmer, D. C. Mattfeld, and N. Soeffker. *Dynamic multi-period vehicle routing: approximate value iteration based on dynamic lookup tables*. Available online at http://web.winforms.phil.tu-bs.de/paper/ulmer/WP_Ulmer_Dynamic_Lookup.pdf. 2016.

[259] M. W. Ulmer and B. W. Thomas. *Enough waiting for the cable guy - estimating arrival times for service vehicle routing*. Available online at http://web.winforms.phil.tu-bs.de/paper/ulmer/Ulmer_SID.pdf. 2017.

[260] M. W. Ulmer. *Approximate Dynamic Programming for Dynamic Vehicle Routing*. Springer International Publishing, 2017.

[261] P. Van Hentenryck, R. Bent, and C. Coffrin. "Strategic Planning for Disaster Recovery with Stochastic Last Mile Distribution." In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Ed. by A. Lodi, M. Milano, and P. Toth. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 318–333.

[262] A. D. Vareias, P. P. Repoussis, and C. D. Tarantilis. "Assessing Customer Service Reliability in Route Planning with Self-Imposed Time Windows and Stochastic Travel Times." *Transportation Science* Articles in Advance (2017).

[263] P. Vayanos, D. Kuhn, and B. Rustem. "Decision rules for information discovery in multi-stage stochastic programming." In: *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*. 2011, pp. 7368–7373.

[264] J. D. L. Vega, P. Munari, and R. Morabito. "Robust optimization for the vehicle routing problem with multiple deliverymen." *Central European Journal of Operations Research* (2017).

[265] B. Vercammen. "Improving the planning accuracy and route efficiency at a freight distribution company: A case study at Van Opzeeland." MA thesis. Eindhoven University of Technology: Department of Industrial Engineering & Innovation Sciences, 2016.

[266] X. Wang and A. Regan. "Assignment Models for Local Truckload Trucking Problems with Stochastic Service Times and Time Window Constraints." *Transportation Research Record: Journal of the Transportation Research Board* 1771 (2001), pp. 61–68.

[267] M. Wen, J.-F. Cordeau, G. Laporte, and J. Larsen. "The dynamic multi-period vehicle routing problem." *Computers & Operations Research* 37.9 (2010), pp. 1615–1623.

[268] W. Wiesemann, D. Kuhn, and B. Rustem. "Robust resource allocations in temporal networks." *Mathematical Programming* 135.1 (2012), pp. 437–471.

[269] W. Wiesemann, D. Kuhn, and M. Sim. "Distributionally Robust Convex Optimization." *Operations Research* 62.6 (2014), pp. 1358–1376.

[270] R. T. Wong. "Vehicle Routing for Small Package Delivery and Pickup Services." In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Springer US, 2008, pp. 475–485.

[271] C. A. Woodward, J. Abelson, S. Tedford, and B. Hutchison. "What is important to continuity in home care?: Perspectives of key stakeholders." *Social Science & Medicine* 58.1 (2004), pp. 177 –192.

[272] Z. Xiang, C. Chu, and H. Chen. "The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments." *European Journal of Operational Research* 185.2 (2008), pp. 534–551.

[273] H. Yaman. "Formulations and Valid Inequalities for the Heterogeneous Vehicle Routing Problem." *Mathematical Programming* 106.2 (2006), pp. 365–390.

[274] N. E. Young. "Sequential and parallel algorithms for mixed packing and covering." In: *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. 2001, pp. 538–546.

[275] B. Zeng and L. Zhao. "Solving Two-Stage Robust Optimization Problems Using a Column-and-Constraint Generation Method." *Operations Research Letters* 41.5 (2013), pp. 457–561.

[276] B. Zeng and Y. An. *Solving Bilevel Mixed Integer Program by Reformulations and Decomposition*. Available at Optimization Online: http://www.optimization-online.org/DB_HTML/2014/07/4455.html. 2014.

[277] C. Zhang, G. Nemhauser, J. Sokol, M.-S. Cheon, and D. Papageorgiou. *Robust Inventory Routing with Flexible Time Window Allocation*. Available at Optimization Online: http://www.optimization-online.org/DB_HTML/2015/01/4744.html. 2015.

[278] Y. Zhang, R. Baldacci, M. Sim, and J. Tang. "Routing optimization with time windows under uncertainty." *Mathematical Programming* (2018).

[279] C. Zhao, J. Wang, J.-P. Watson, and Y. Guan. "Multi-Stage Robust Unit Commitment Considering Wind and Demand Response Uncertainties." *IEEE Transactions on Power Systems* 28.3 (2013), pp. 2708–2717.

[280] L. Zhao and B. Zeng. *An Exact Algorithm for Two-stage Robust Optimization with Mixed Integer Recourse Problems*. Tech. rep. University of South Florida, 2012.